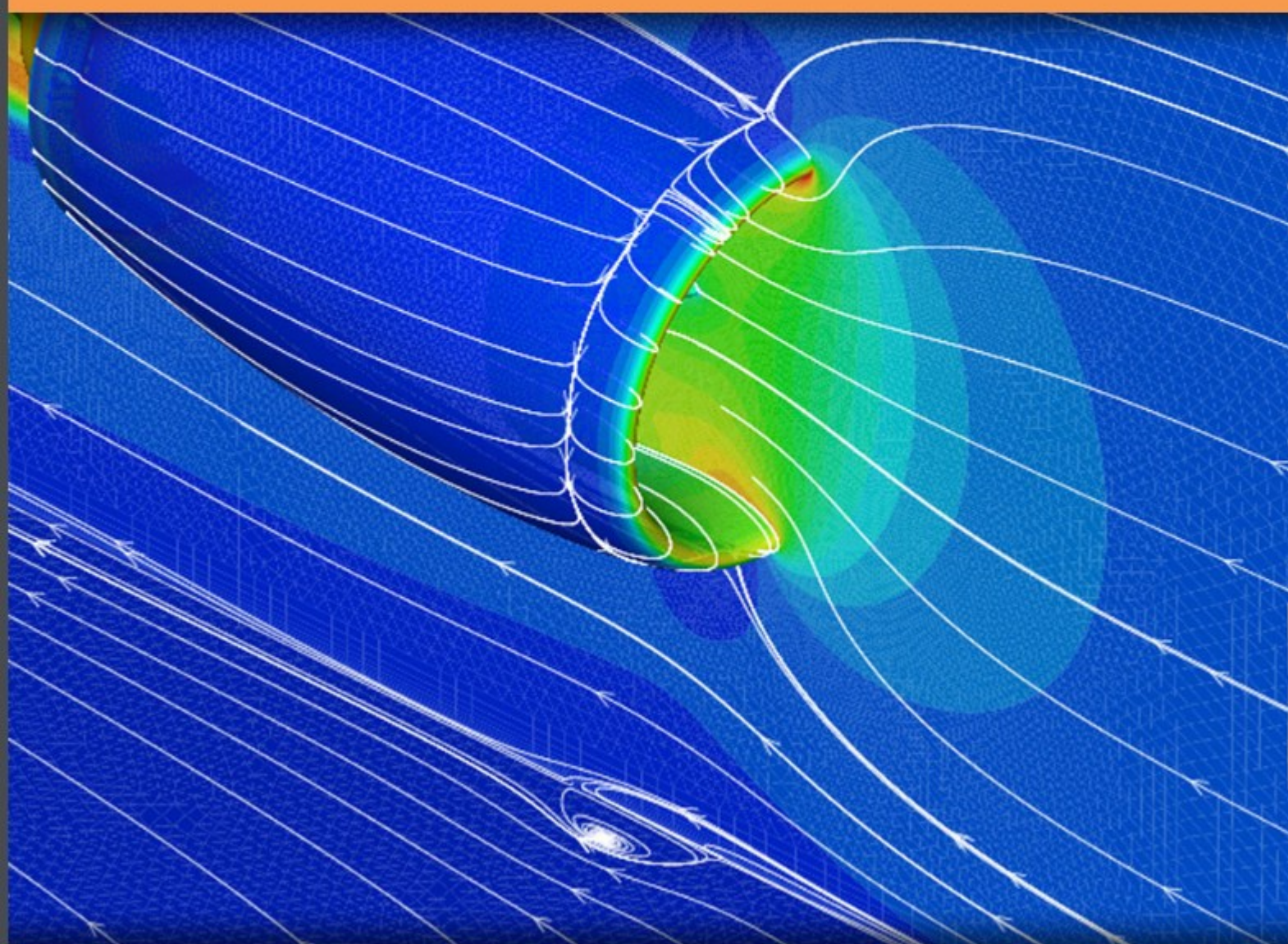
bookboon.com

Computational Fluid Dynamics

Abdulnaser Sayma

Abdulnaser Sayma

Computational Fluid Dynamics

Contents

1 Introduction

Computational Fluid Dynamics (CFD) is the branch of fluid dynamics providing a cost-effective means of simulating real flows by the numerical solution of the governing equations. The governing equations for Newtonian fluid dynamics, namely the Navier-Stokes equations, have been known for over 150 years. However, the development of reduced forms of these equations is still an active area of research, in particular, the turbulent closure problem of the Reynolds-averaged Navier-Stokes equations. For non-Newtonian fluid dynamics, chemically reacting flows and two phase flows, the theoretical development is at less advanced stage.

Experimental methods has played an important role in validating and exploring the limits of the various approximations to the governing equations, particularly wind tunnel and rig tests that provide a cost-effective alternative to full-scale testing. The flow governing equations are extremely complicated such that analytic solutions cannot be obtained for most practical applications.

Computational techniques replace the governing partial differential equations with systems of algebraic equations that are much easier to solve using computers. The steady improvement in computing power, since the 1950’s, thus has led to the emergence of CFD. This branch of fluid dynamics complements experimental and theoretical fluid dynamics by providing alternative potentially cheaper means of testing fluid flow systems. It also can allow for the testing of conditions which are not possible or extremely difficult to measure experimentally and are not amenable to analytic solutions

1.1 Scope of this book

There is a large number of commercial CFD packages in the market nowadays and CFD has established itself as a useful analysis and design tool. In addition, there is a large number of research and public domain CFD programmes. As a student you are most likely to use an existing CFD programme than write a new one from scratch. In some occasions, students might do certain modifications or additions to existing programmes to tailor them for a particular problem.

On the other hand there is a large number of published CFD books. However most of those, if not all, are targeted towards postgraduate students or researchers who are interested in understanding, in detail, the numerical algorithms to enable them to develop or adapt CFD programs.

Commercial CFD tools usually come with user’s guides and examples manuals that provide users with information of how to use that particular tool. However, in most cases, there is no explanation of the theoretical background which enables the user to make an informed choice of the technique used, or the type of boundary conditions to apply.

This book aims at bridging the gap between the two streams above by providing the reader with the theoretical background of basic CFD methods without going into deep detail of the mathematics or numerical algorithms. This will allow students to have a grasp of the basic models solved, how they are solved and the reasoning behind the choice of any particular method. This will give them an informed choice when they want to apply CFD tools to a particular engineering problem.

Thus the rest of this Chapter will present an overview of engineering prediction methods comparing the scope, advantages and limitations of experimental methods, analytical methods and CFD techniques. It will then present typical problems that can be solved by CFD for illustration purposes. It will then end with outlining the structure of the rest of this book to help the student find his way through.

1.2 Prediction Methods

Engineers are interested in predicting the behaviour of systems to understand the relationship between the system variables. This allows for better design of systems or understanding of their behaviour for optimising their operation. Typically, engineers used to perform experiments which either allows them to understand the system directly, or construct mathematical models that represent their systems.

Another approach to understand the system is to construct a mathematical model based on the understanding of the basic physical phenomena that govern its behaviour and then trying to solve these models for a given set of conditions by finding a mathematical solution to the resulting system of equations. This is termed the analytical approach.

The third approach is the use of CFD methods mentioned above, where the differential equations governing the system are converted to a set of algebraic equations at discrete points, and then solved using digital computers. We will now shed some light on these three approaches highlighting their advantages and limitations.

1.1.1 Experimental Techniques

The most reliable information about physical phenomena is usually given by measurements. In certain situations, an experimental investigation involving full-scale equipment can be used to predict how the equipment would perform under given conditions. However, in most practical engineering applications, such full scale tests are either difficult or very expensive to perform, or not possible at all.

A common alternative is to perform experiments on small scale models. The resulting information however, needs to be extrapolated to the full scale and general rules for doing this are often unavailable. The small scale models do not usually simulate all the features of the full scale system. This sometimes limits the usefulness of the test results.

In many situations, there are serious difficulties in measurements and the measuring equipment can have significant errors. For example, the performance of an aircraft engine at high altitude conditions is a difficult, expensive and sometimes a risky undertaking, and is usually done at the later stages of the process where major changes to the design can result in significant costs.

Although the above discussion implies that the need for reliable computational models is of paramount importance, it is should be stressed that these numerical models require validation using reliable experimental data before they can be put to good use. This indicates that experimental methods will remain to play an important rule in engineering.

1.1.2 Analytical methods

Analytical models work out the consequences of a mathematical model which represents the behaviour of a system. The mathematical model representing the physical process mainly consists of a set of differential equations. If classical mathematics were used to solve these equations, we call the approach as the analytical or theoretical approach.

In most practical engineering applications, various assumptions and simplifications need to be made to enable the analytical solution of the differential equations representing the physical situation. This at one hand limits the applicability of these methods to simple type problems, or limits the validity of the solutions if too many assumptions and simplifications are made.

Despite that, analytical methods played significant role in the past and they still play an important role. They have helped engineers and scientists in the understanding of the fundamental rules controlling the behaviour of many engineering systems. In addition, they are used to help understand and interpret experimental results. Furthermore, they can be used as a first stage in the validation of CFD models.

1.1.3 CFD techniques

CFD techniques have emerged with the advent of digital computers. Since then, a large number of numerical methods were developed to solve flow problems using this approach. The basic approach is outlined below.

The purpose of a flow simulation is to find out how the flow behaves in a given system for a given set of inlet and outlet conditions. These conditions are usually termed boundary conditions.

For example, in a boiler required to raise the temperature of water for heating purposes, it may be required to calculate, for a given mass inflow of water and energy input using the gas fire, what is the temperature and velocity of the water coming out of the boiler. It might be also required to know the flow pattern and temperature distribution within the boiler if design improvements need to be made to improve mixing or reduce energy loss through the walls.

Since the geometry in most boilers is complex, it is difficult to find an analytical solution to the flow equations. For all engineering purposes, it will be useful to know the basic flow quantities at a large number of discrete points spread around the boiler geometry. This will give enough understanding of the flow behaviour and will enable engineers to get the required information either for operation or design purposes.

The basic concept of CFD methods is then to find the values of the flow quantities at a large number of points in the system. These points are usually connected together in what is called numerical grid or mesh. The system of differential equations representing the flow is converted, using some procedure, to a system of algebraic equations representing the interdependency of the flow at those points and their neighbouring points.

The resulting system of algebraic equations, which can be linear or non-linear, is usually large and requires a digital computer to solve. In essence, we end up with a system with the unknowns being the flow quantities at the grid points. Solution of this system results in the knowledge off these quantities at the grid points.

If the flow is unsteady, either due to varying boundary conditions, or due to inherent unsteadiness. The solution procedure is repeated at discrete time intervals to predict the evolution in time of the flow variables at the grid points.

With the development of fast and validated numerical procures, and the continuous increase in computer speed and availability of cheap memory, larger and larger problems are being solved using CFD methods at cheaper cost and quicker turn around times. In many design and analysis applications, CFD methods are quickly replacing experimental and analytical methods.

It should be noted that there are certain levels of numerical approximations and assumptions made during the development of CFD models. Hence, good understanding of the applicability range and the limitation of a CFD tools is essential to enable the correct use of these tools.

In addition to the speed and reduced cost of CFD methods, compared to experimental procedures in most engineering applications, they also offer a more complete set of information. They usually provide all relevant flow information throughout the domain of interest. Experimental methods are mostly limited to measurements of some flow quantities at certain locations accessible by the measuring equipment.

CFD simulations also enable flow solutions at the true scale of the engineering systems with the actual operating conditions, while experimental measurements mostly require either scaling up or down. In most cases, realistic conditions cannot be economically represented and thus results need to be extrapolated. This problem does not exit in CFD simulations.

1.3 Typical problems

In this Section, typical problems solved using CFD techniques are presented for illustration purposes, where advantages and limitations of CFD methods are highlighted. These will be explained in the light of the understanding developed throughout this book.

1.3.1 Aerospace applications

CFD methods are now widely used in most aerospace applications for the purpose of predicting component performance and as an integral part of the design cycle. The applications are numerous and we will only list few examples here.

The first example is flow around an aircraft. Wind tunnel tests require substantial scaling which leads to some difficulties of matching the important flow parameters. If we attempt to model the correct Mach number, the Reynolds number will be substantially lower than the full scale Reynolds number leading to errors in the modelled shear stress and other flow features. It is also very expensive to replicate altitude conditions within a wind tunnel.

On the other hand, full scale flight tests are extremely expensive and are not without risk. For these reasons, CFD provides a useful tool in predicting the performance of the airframe components under various conditions and this leads to substantial cuts in the time and cost of the design process. An example of a flow around a complete aircraft is shown in Figure 1.1.

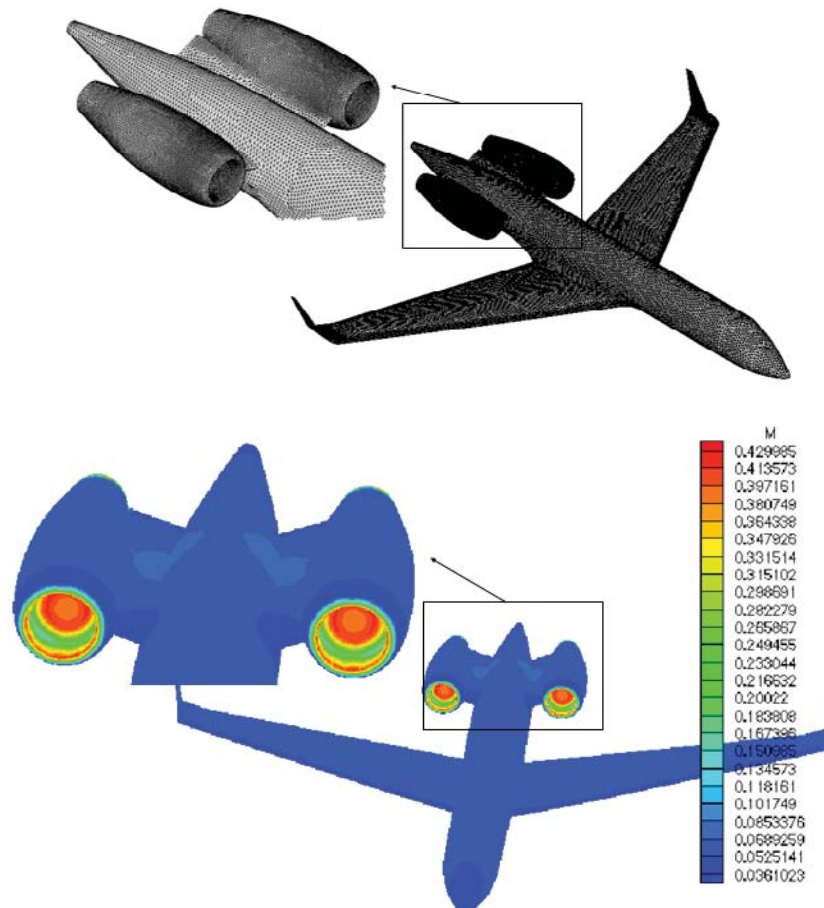



Figure 1-1: Grid and flow solution for a civil aircraft with nacelles

Within the aircraft engine CFD can be used for modelling the flow in the primary flow path which includes turbine and compressor blades to secondary flow path showing cooling cavities, bleeds, around seals and also for intake ducts. Some examples are shown in Figure 1.2.

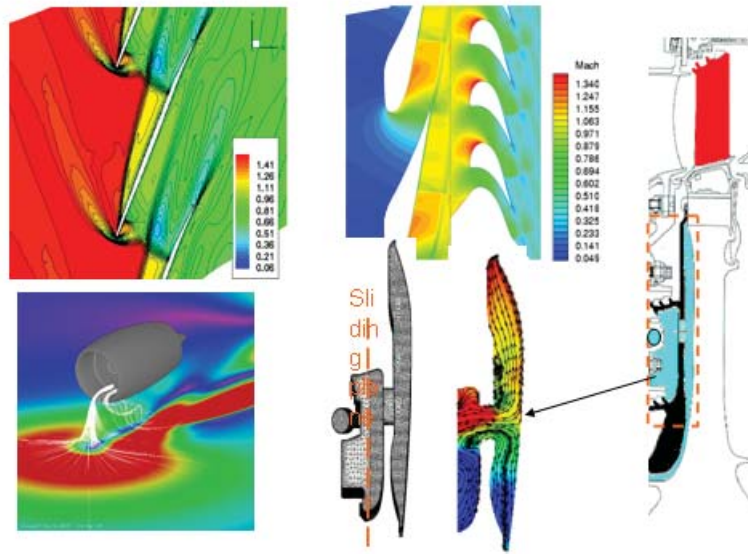



Figure 1-2: Various turbomachinery applications

1.3.2 Automotive applications

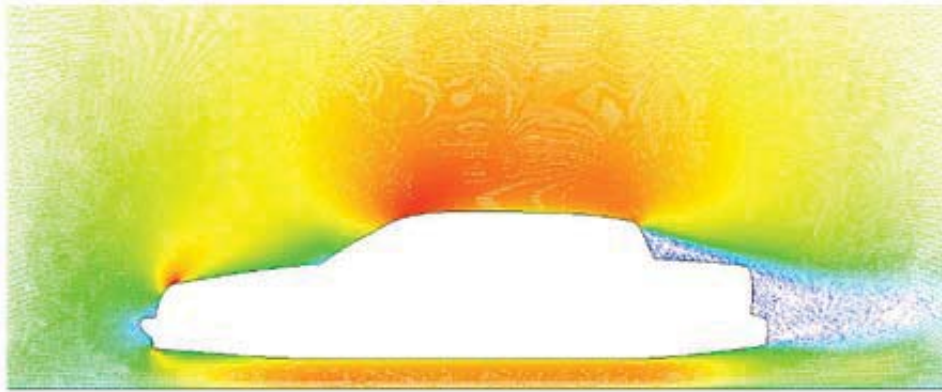
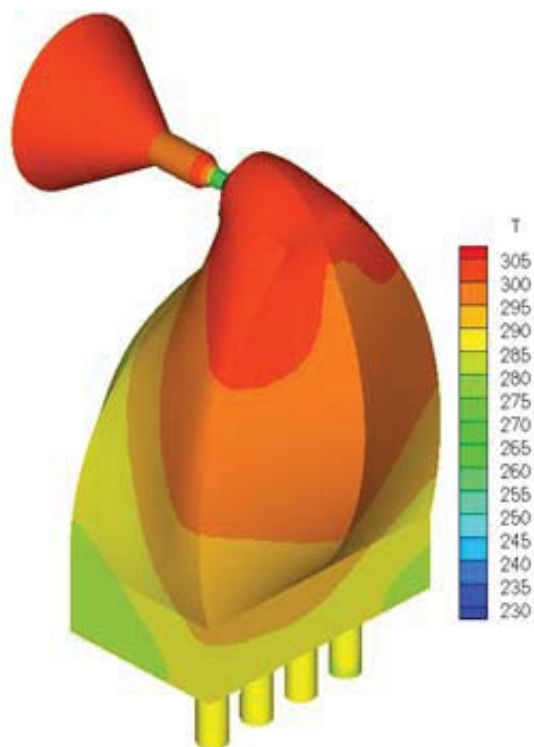
In automotive applications CFD is nowadays used in a large number of areas including engine components, auxiliary systems and also for modeling the aerodynamics of the car to minimise drag and optimise the down force under various operating conditions. Figure 1.3 shows two examples of automotive applications. Figure 1.3a shows the flow field around a family car obtained using CFD methods. Figure 1.3b shows the flow in the induction duct for a formula student racing car. The objectives of this analysis are to understand the flow pattern within the induced during the process of design optimisation to minimise pressure losses and ensure uniformity of air ducted to the cylinders.

(a) Flow around a car

(b) Formula student racing car induction duct

Figure 1-3: Examples of automotive applications

1.3.3 Biomedical applications

In bio-medical applications, CFD is nowadays used to model the flow of blood in the heart and vessels the flow in heart assist devices and various other fluid flow equipment such as drug and anaesthetic devices and inhalers. The use if CFD reduces the need for tests on human being and animals until the last stage of the process. Figure 1.4 shows an example of modelling air flow in an inhaler. The modelling process is used to optimise the design of the inhaler to enhance mixing of the drug dose and inhaled air while keeping high discharge coefficient to minimize patient suction.

Figure 1-4: Grid and flow in inside an inhaler

1.4 Outline of this book

Since CFD methods are concerned with the solution of the differential equations representing flow, Chapter 2 will present an overview of the basic flow equations and the physical interpretation of the terms in those equations. It will also present the different levels of approximation to the flow equations which are typically used as a basis for the development of numerical models.

It is advisable to study this Chapter carefully as it presents the basic grounds the numerical models will be developed from. It will also give the general terminology of the flow models used in this book. If you think that you understand the basic flow equations, you can jump directly to Chapter 3.

Chapter 3 will present an overview of the basic discretisation techniques used to convert the continuum differential equations representing the flow, to a discrete form which is amenable for solution using digital computers. Illustrative examples will be given to simplify the understanding.

Chapter 4 will focus on the basic properties of numerical schemes which make them valid for the solution of the flow equations. It will also help understand some of the terminology used in solvers that helps the student to make an informed choice when he selects a particular method.

Chapter 5 will describe in more detail the Finite Difference Method, which is the simplest procedure used to convert the differential equations to discrete form.

Chapter 6 will describe the Finite Element method which was developed at first for structural dynamics problems. It was then taken by CFD developers to facilitate solving more complex geometry problems that are difficult to discretise using Finite Difference method.

Chapter 7 will describe the Finite Volume Method, which is nowadays, the most popular method for CFD. This method can be viewed as a subset of the Finite Element methods.

Chapter 8 focuses on the solution methods for the systems of algebraic equations resulting from the numerical discretisation process.

Chapter 9 will give a brief account of the various grid, or mesh, generation procedures. It aims at helping the student choose the appropriate mesh generation procedure for a particular problem and to understand the advantages and limitations of each method.

2. Basic Equations of Fluid Flow and Levels of Approximation

Fluid flow equations are made essentially of differential equations representing the interrelationship between the flow variables and their evolution in time and space. These equations are complemented by algebraic relations such as the equation of state for compressible flow as we will see below.

To help the student understand the physical meaning of the terms in these equations, we will illustrate the basic concepts using a simple differential equation before presenting the system of differential equations representing the flow.

Let us take for example, the temperature distribution of a flow in straight a pipe, where the velocity is fixed by pumping a fixed volume flow rate into the pipe, Assume also that the flow velocity is not altered by the change in temperature. The temperature distribution $T(x,t)$ in the pipe as a function of the pipe axial coordinate x is given by the following differential equation:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} - \alpha\frac{\partial^2 T}{\partial x^2} = 0 \qquad (2.1)$$

where u is the flow velocity which is assumed constant across the pipe, t is time and α is the thermal diffusivity. In Equation 2.1, the first term is the time derivative expressing the temperature gradient with time. The second term is called the advection term which is responsible for the transport of any temperature disturbance with the flow without any distortion.

The third term is called the diffusion term, which is responsible for the spread of any disturbance in all directions. To illustrate these concepts, let us take these terms one by one. If the thermal diffusivity is assumed to be negligible, Equation 2.1 reduces to:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0 \qquad (2.2)$$

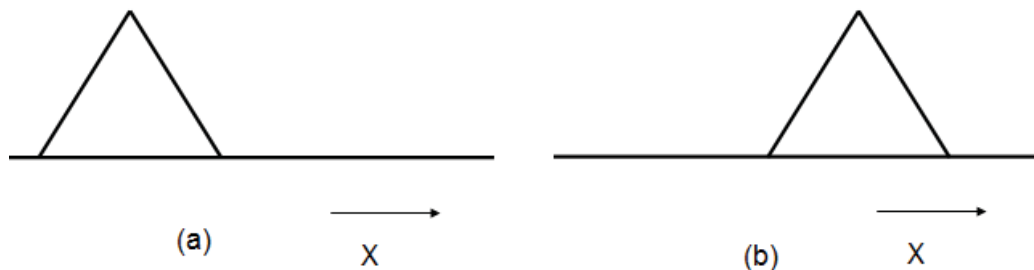



Figure 2.1: Convection of a disturbance in a pipe

A temperature disturbance entering the pipe from the left (Figure 1(a)) at time t = 0 will be convected without any distortion after a while to the right as shown in Figure 1(b). If the flow velocity is zero, Equation 2.1 reduces to:

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0 \qquad (2.3)$$

A temperature disturbance shown in Figure 2.2(a) will be diffused to the right and left and reduced in amplitude after a while as shown in Figure 2.2(b). The action of both convection and diffusion can now be added together to represent the behaviour described by equation 2.1 as shown in Figure 2.3, where a disturbance introduced at the left, will both be convected and diffused as shown in Figure 2.3(b).

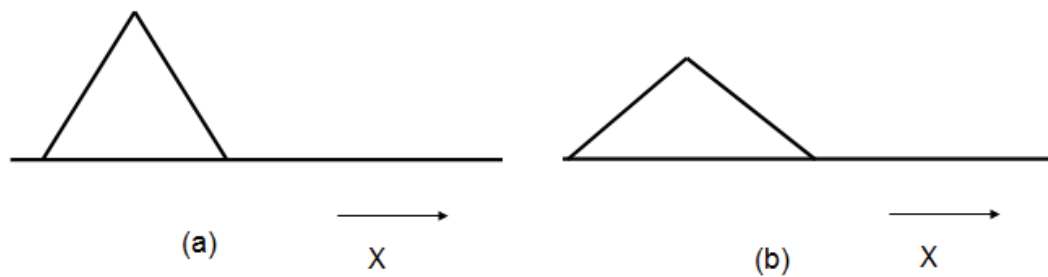



Fig 2.2: Diffusion of a wave in a pipe

The simplicity of this system enabled a straightforward interpretation of the various terms. Equation 2.1 is said to be linear because the velocity is assumed constant and does not depend on temperature. Hence there is only one dependent variable, which is the temperature.

A system which might include a nonlinear convective term is shown in the equation below:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \qquad (2.4)$$

This equation represents the transport of momentum along the pipe. In this case, velocity is the dependent variable and it is not assumed to be constant. The nonlinearity arises in the convective term where the velocity, which is the sought variable, is being convected by the action of the velocity itself.

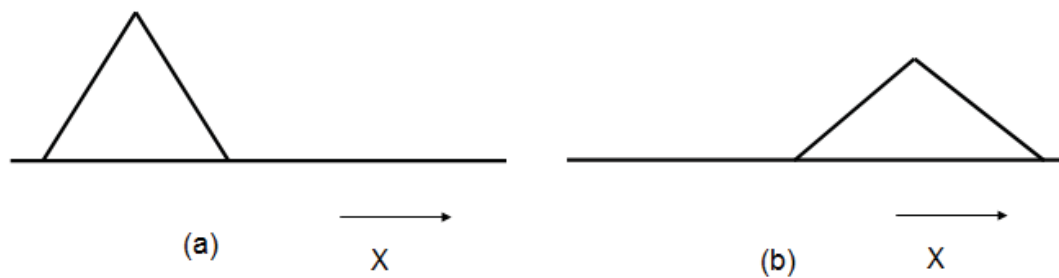
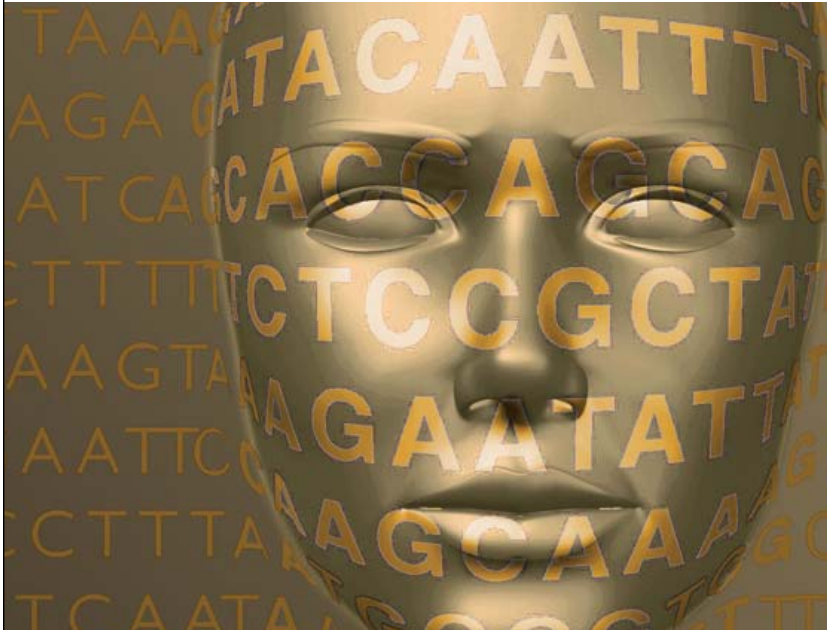



Fig 2.3: Convection and diffusion of a wave in a pipe

2.1 The Navier-Stokes Equations

The Navier-Stokes and continuity equations provide the foundations for modeling fluid motion.

The laws of motion that apply to solids are valid for all matter including liquids and gases. A principal difference, however, between fluids and solids is that fluids distort without limit. For example, if a shear stress is applied to a fluid, then layers of fluid particles will move relative to each other and the particles will not return to their original position if application of the shear force is stopped. Analysis of a fluid needs to take account of such distortions.

A fluid particle will respond to a force in the same way that a solid particle will. If a force is applied to a particle, acceleration will result as governed by Newton's second law of motion, which states that the rate of change of momentum of a body is proportional to the unbalanced force acting on it and takes place in the direction of the force. It is useful to consider the forces that a fluid particle can experience. These include:

body forces such as gravity and electromagnetism;
forces due to pressure;
forces due to viscous action;
forces due to rotation.

Assuming that the shear rate in a fluid is linearly related to shear stress, and that the fluid flow is laminar, Navier (1823) derived the equations of motion for a viscous fluid from molecular considerations. Stokes (1845) also derived the equations of motion for a viscous fluid in a slightly different form and the basic equations that govern fluid flow are now generally known as the Navier-Stokes equations of motion. The Navier-Stokes equations can also be used for turbulent flow, with appropriate modifications.

The Navier-Stokes equations can be derived by considering the dynamic equilibrium of a fluid element.

They state that the inertial forces acting on a fluid element are balanced by the surface and body forces

We are not going to derive the Navier-Stokes and continuity equations here as this can be found in most standard text books. However, we will state the equations here and briefly give the physical interpretation of the terms as this will help us understand the numerical schemes used to solve those equations. It will also allow us to introduce the various levels of approximations used to simplify the equations to reduce the numerical solution costs.

2.1.1 Compressible flow

The flow governing equations are the continuity equation, momentum equation (Navier-Stokes) and energy equation:

The continuity equation:

$$-\frac{\partial \rho}{\partial t} = \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} \quad (2.5)$$

The Navier Stokes Equation:

$$\underbrace{\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)}_{Inertial\ terms}=-\underbrace{\frac{\partial p}{\partial x}}_{\text{Pressure } gradient}+\underbrace{\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)}_{Viscous\ terms}+\underbrace{F_x}_{Body\ force\ terms} \quad (2.6)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+F_y \quad (2.7)$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+F_z \quad (2.8)$$

The energy equation

$$\rho c_p\left(\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}+w\frac{\partial T}{\partial z}\right)=\Phi+\frac{\partial}{\partial x}\left[k\frac{\partial T}{\partial x}\right]+\frac{\partial}{\partial y}\left[k\frac{\partial T}{\partial y}\right]+\frac{\partial}{\partial z}\left[k\frac{\partial T}{\partial z}\right]$$
$$+\left(u\frac{\partial p}{\partial x}+v\frac{\partial p}{\partial y}+w\frac{\partial p}{\partial z}\right) \quad (2.9)$$

where Φ is the dissipation function given by:

$$\Phi=2\mu\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial w}{\partial z}\right)^2+0.5\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2+0.5\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)^2+0.5\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)^2\right]$$
$$-\frac{2}{3}\mu\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)^2$$

In these equations, v, v, w are the velocity components in the x, y, z directions, ρ is the density, T is the temperature, p is the pressure, μ is the viscosity and c_p is the specific heat at constant pressure.

The continuity equation is an expression representing that matter is conserved in a flow. Per unit volume, the sum of all masses flowing in and out per unit time must be equal to the change of mass due to change in density per unit time.

The continuity equation applies to all fluids, compressible and incompressible flow, Newtonian and non-Newtonian fluids. It expresses the law of conservation of mass at each point in a fluid and must therefore be satisfied at every point in a flow field.

It is worthwhile to offer brief comment on the physical significance of Equations (2.6-2.8). The terms on the left side are often referred to as inertial terms, and arise from the momentum changes. These are countered by the pressure gradient, $\partial p / \partial x$, viscous forces which always act to retard the flow, and if present, body forces.

The inertial term gives a measure of the change of velocity of one fluid element as it moves about in space. The term $\partial / \partial t$ gives the variation of velocity at a fixed point and is known as the local derivative. The remaining three terms of the inertial term are grouped together and known as the convective terms or convective differential.

Assuming constant properties of viscosity and specific heat, the above system of equations contains 6 unknowns. With only five equations available, a further equation is needed to close the system. Usually, this is provided by a constitutive relation for the pressure. For example, for an ideal gas, the relation between temperature and pressure is given by: $p = \rho RT$, where R is the gas constant.

2.1.2 Incompressible flow

The above system of equation can be simplified if the density is constant. If the temperature is also is assumed constant, the system reduces to (for simplicity, body forces are also neglected, but they can be retained if needed):

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \qquad (2.10)$$

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \qquad (2.11)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \qquad (2.12)$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) \qquad (2.13)$$

For incompressible flows with temperature variation, the energy equation need to be solved to obtain the temperature variation, but its effect on the continuity and momentum equations are neglected.

2.2 Turbulent flow

In the 1880's, Osborne Reynolds carried out his historic visualisation studies of flow in a pipe. He observed that well-ordered laminar flow degenerated into a chaotic motion when the velocity in the pipe reached a certain value. This was linked to a non-dimensional quantity called the flow Reynolds number $\mathrm{Re}=\frac{\rho UD}{\mu}$ where U is the average velocity in the pipe and D is the diameter.

The Reynolds number represents the ratio between inertia forces and viscous forces. If this number is low, the flow is orderly with parallel streamlines. If it is increased, at some point, this structure of laminar flow loses its identity, giving rise to a flow structure characterised by large-scale eddies.

Generally speaking, viscous effects, and consequently turbulence, prevail in a region close to solid boundaries called the boundary layer. In pipe flows, the boundary layer grows as flow from, say, a plenum until it covers the whole pipe.

For external flows, such as flow over a wing or a car, the boundary layer is confined to a narrow region close to the wall. Away from the wall viscous effects are negligible and the flow is termed inviscid.

To understand how the boundary layer forms, imagine flow with free stream velocity U_∞ approaching a flat plate as shown in Fig 2.4. The flow will have no slip or zero velocity at the wall due to friction. At a distance far from the wall, the flow has a velocity U_∞. As the flow approaches the wall, a boundary layer forms where the flow varies from zero at the wall to U_∞ at the far stream.

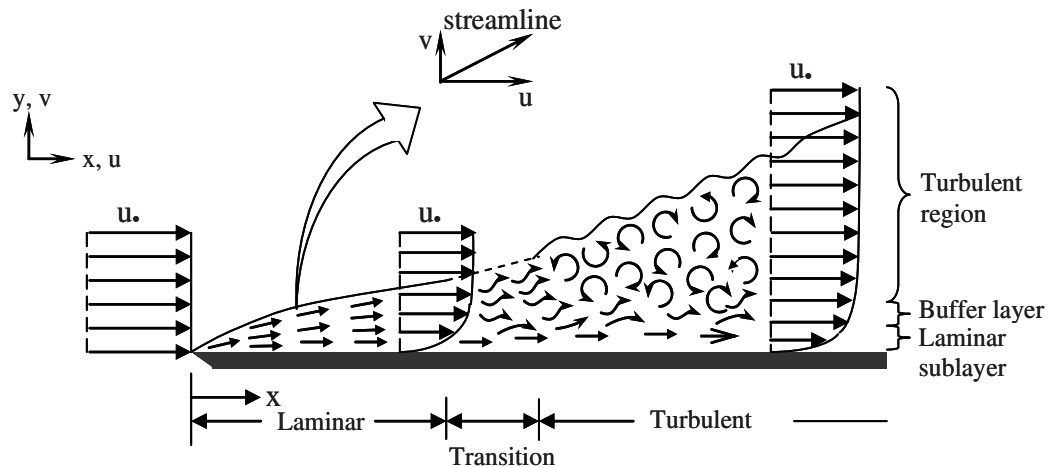



Fig 2.4 Boundary layer over a flat plate

The boundary layer starts as laminar as the Reynolds number $\mathrm{Re} = \dfrac{\rho U x}{\mu}$ is small, which means that inertia forces are still small compared to viscous forces. As x increases along the blade, the Reynolds number increases, and inertial forces start to dominate causing instability in the boundary layer to develop. A transition zone occurs until the flow develops to fully turbulent flow with chaotic eddies.

Underneath the turbulent boundary layer, a region close to the wall remains laminar and this is called the laminar sublayer. We will now focus our attention on the turbulent nature of the flow and its implications on modelling.

Turbulence is of fundamental interest to engineers because most flows encountered in engineering are turbulent. This some times happens because it is difficult to keep the flow laminar, or by intention as turbulence is essential for the engineering application. For example, in heat exchangers, turbulence enhances mixing and thus improves heat transfer coefficient.

Consider a steady turbulent flow situation where velocity is measured at a point with a device that records the variation with time. A signal similar to that shown in Figure 2.5 is obtained. A different signal will be obtained for each point in space. For highly turbulent flows, the variation in time is so random and its detailed variation can be of little, if any, engineering relevance. It is the average quantity that is useful for most engineering applications.

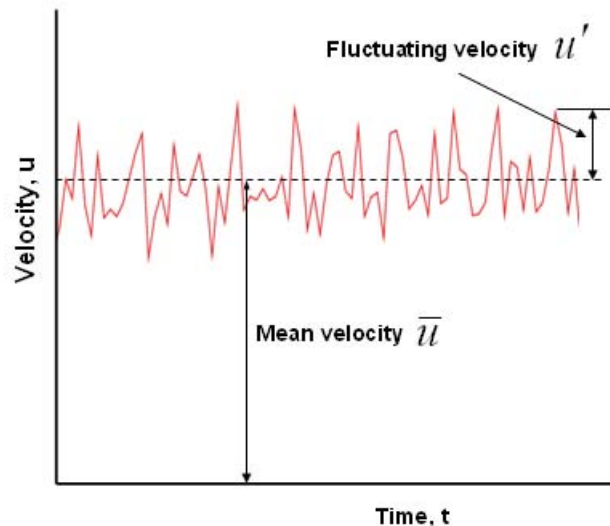



Figure 2.5 Instantaneous and average velocity in turbulent flow

The average, or mean value of the velocity, or any other flow quantity, can be obtained by time averaging over an interval large enough to filter out small scale variations, but small enough so that meaningful large scale time variations are maintained. For the chosen time interval Δt , the time average velocity is obtained using the relation:

$$\bar{u} = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} u\,dt \qquad (2.14)$$

where:

$$u = \bar{u} + u' \qquad (2.15)$$

This averaging can similarly be applied to all other flow variables thus we can define $v = \bar{v} + v'$, $p = \bar{p} + p'$ and so on.

The use of this averaging process is not only useful for engineering applications directly, but also for numerical modelling. The reason is that if we are to build numerical model to resolve the instantaneous quantities, with their spatial and temporal variations, we end up with very fine grids and very small time steps that are impractical, if possible, to solve.

The averaged quantities require coarser grids and larger time steps making CFD computations easier to achieve. This is useful also bearing in mind that these average quantities are mostly what we need for engineering purposes.

If we substitute the instantaneous quantities of the flow equations by the sum of the average and fluctuating components as in Equation 2.15 into the flow equations describing the flow and by taking the time average of the terms in the equations, we obtain what is known as the Reynolds-averaged flow equations.

The derivation of the Reynolds-averaged Navier-Stokes equations can be found in most fluid mechanics text books. See for Example Schlichting and Gersten (2000). It is sufficient for the purpose of this book to state the final outcome of this derivation and explain its implications. To simplify the presentation, we will present it for the two dimensional steady-flow.

The two dimensional steady state incompressible flow equations can be stated as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad (2.16)$$

$$\rho\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right) \qquad (2.17)$$

$$\rho\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right) \qquad (2.18)$$

The corresponding Reynolds-averaged equations are:

$$\frac{\partial \overline{u}}{\partial x}+\frac{\partial \overline{v}}{\partial y}=0 \tag{2.19}$$

$$\rho\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial \overline{p}}{\partial x}+\frac{\partial}{\partial x}\left(\mu\frac{\partial \overline{u}}{\partial x}-\rho u'u'\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial \overline{u}}{\partial y}-\rho u'v'\right) \tag{2.20}$$

$$\rho\left(\overline{u}\frac{\partial \overline{v}}{\partial x}+\overline{v}\frac{\partial \overline{v}}{\partial y}\right)=-\frac{\partial \overline{p}}{\partial y}+\frac{\partial}{\partial x}\left(\mu\frac{\partial \overline{v}}{\partial x}-\rho u'v'\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial \overline{v}}{\partial y}-\rho v'v'\right) \tag{2.21}$$

We immediately notice the similarity between the expressions of Equations 2.16-2.18 to those of Equations 2.19-2.21. There are two differences however. All the instantaneous quantities were replaced by their corresponding time-averaged quantities. The second noticeable difference is in the momentum equations, where the extra terms $-\rho u'u'$, $-\rho v'v'$ and $-\rho u'v'$ appear.

These terms behave like stress terms, the first two are normal stresses and the last is a shear stress term. These additional terms represent the effect of the fluctuating velocity components on the mean velocity field. They are usually called Reynolds-stresses. In three dimensional flows, 6 of these terms appear in the equations. These terms are not known and require further equations if the system is to be solved.

2.2.1 Turbulence modelling

The process of finding a closure to the above system of equations is called turbulence modelling. We will outline two approaches which are commonly used in CFD. The first is called second moment closure and the second and more popular one is called the eddy-viscosity or turbulent viscosity approach.

In the second moment closure approach, differential equations, or what is called transport equations are derived for each of the Reynolds stresses. These differential equations are derived by manipulations of the Navier-Stokes equations and introducing approximate models for complex terms in terms of the flow gradients.

The resulting equations are similar to the Navier-Stokes equations in the sense that they have a time derivative term, convective terms, diffusion terms and source terms. A further term usually appears in the equations called the dissipation term which requires another differential equation to describe its transport.

In three dimensional incompressible flows, the system results in seven additional differential equations for the six Reynolds-Stress terms and the dissipation term. These need to be solved together with the four flow equations to get a complete solution for the flow field. Obviously this represents a large amount of computational effort. What makes its even more time consuming is that the Reynolds-Stress equations are usually stiffer and take more time to converge than the actual flow equations.

An alternative approach to close the system was proposed by Prandtl, which may be less accurate, but more practical. This is the so called the eddy viscosity approach. In this approach, it was hypothesised the Reynolds stress terms behave like the viscous terms and a new quantity called the turbulent viscosity was introduced such that:

$$-\rho u'u' = \mu_t \frac{\partial \bar{u}}{\partial x}, \quad -\rho v'v' = \mu_t \frac{\partial \bar{v}}{\partial x}, \quad -\rho u'v' = \mu_t \frac{\partial \bar{u}}{\partial y} = \mu_t \frac{\partial \bar{v}}{\partial x} \tag{2.22}$$

If Equation 2.22 is substituted into the Reynolds-averaged momentum Equations 2.20 and 2.21, we obtain:

$$\rho\left(\bar{u}\frac{\partial \bar{u}}{\partial x} + \bar{v}\frac{\partial \bar{u}}{\partial y}\right) = -\frac{\partial \bar{p}}{\partial x} + \frac{\partial}{\partial x}\left((\mu + \mu_t)\frac{\partial \bar{u}}{\partial x}\right) + \frac{\partial}{\partial y}\left((\mu + \mu_t)\frac{\partial \bar{u}}{\partial y}\right) \tag{2.23}$$

$$\rho\left(\bar{u}\frac{\partial \bar{v}}{\partial x} + \bar{v}\frac{\partial \bar{v}}{\partial y}\right) = -\frac{\partial \bar{p}}{\partial y} + \frac{\partial}{\partial x}\left((\mu + \mu_t)\frac{\partial \bar{v}}{\partial x}\right) + \frac{\partial}{\partial y}\left((\mu + \mu_t)\frac{\partial \bar{v}}{\partial y}\right) \tag{2.24}$$

This has the same shape as the original momentum equation with the exception that the viscosity is replaced by the sum of the viscosity and the turbulent viscosity. The closure problem becomes that of finding the distribution of μ_t in the solution domain.

Several models of different levels of complexity were developed to find the values of the eddy viscosity. They can be classified according to the number of differential equations solved to obtain this unknown.

The simplest of these forms are called the zero-equation models. In these models, simple algebraic relations are used to obtain the distribution of eddy viscosity. The next order of complexity is using what is called one-equation models. In these models, a transport equation is derived for the transport of either the eddy viscosity, or a related variable where eddy viscosity can be computed.

These transport equations are also similar in nature to the momentum equation in the sense that they contain a time derivative term, convection, diffusion and source and sink terms. The most popular one-equation models are the Spallart and Allmaras (1992) model and Baldwin and Barth model (1990).

A further level of complexity comes from solving two differential, or transport equation. One representing the generation and transport of turbulence and the other represent the transport of dissipation of turbulence. The turbulent viscosity can be computed from the two transported terms in these equations. Examples of two equation turbulence models are the $k-\varepsilon$ and $k-\omega$ (Wilcox, 2004)

2.3 Inviscid flow

In a number of engineering flows, particularly high speed flows, the boundary layer is confined to a very thin region near the wall. Outside that region viscous effects are negligible. This can be also restated as the viscous terms are very small compared to the time derivative and the advection terms in the momentum equations. In this case, it is possible to get a good description of the flow field by removing the viscous terms from the equations. This results in the inviscid flow model.

For example, for incompressible flow, the momentum equations 2.11-2.13 take the form:

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x} \quad (2.25)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y} \quad (2.26)$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z} \quad (2.27)$$

You should be able to notice here that turbulence does not play a role and there is no need to perform Reynolds-averaging on the equations. Hence there is no need for turbulence modelling.

2.4 Boundary layer approximation

It is possible to simplify the flow equations in the boundary layer by making and order of magnitude analysis of the different terms in the momentum equations and neglecting the terms that are very small relative to other terms. The resulting terms apply only in the boundary layer and provide a simpler form of the equations which can be solved easily even sometimes using an analytical approach.

The basic boundary layer assumption, usually used, is that the thickness of the boundary layer is much smaller than the extent that the flow travels in the stream-wise direction. For a flat plate boundary layer where the flow is along the x-axis and the boundary layer thickness is in the y-direction, for example, the two dimensional boundary layer equations are:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \qquad (2.28)$$

$$\rho\left\{u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right\}=-\frac{\partial p}{\partial x}+\mu\frac{\partial^2 u}{\partial y^2}+F_x \qquad (2.29)$$

These equations can be used to derive numerical schemes where boundary layer equations are solved in the boundary layer and the inviscid flow equations of the above Section can be solved in the rest of the domain.

2.5 Closure

This Chapter gave a brief account of the general nature of the differential equations that describe the flow behaviour. Different levels of approximation most common in modern CFD solvers were introduced and the physical interpretation of the terms was give, in addition to a description of the validity range for each approximation.

Turbulent flows were introduced and the Reynolds-Averaging and turbulence modelling were accounted for. Additionally, inviscid flow models and boundary layer approximation were introduced.

It should be noted that this is not a comprehensive account of all possible underlying models or possible approximation. Nevertheless, it covers the most common models and approximations and opens the way for readers to explore further details when needed.

We shall proceed now in describing CFD methods used to solve these systems of equations using digital computers.

3. Basic Computational Techniques

In this Chapter, basic computational techniques will be introduced. We will start first by describing how the typical terms in a fluid flow differential equation, namely first and second derivatives are converted to approximate discrete expressions which can be used in the construction of numerical schemes. We will briefly talk about time discretisation, leaving more details about this later on as further concepts are developed.

A general description of the most common discretisation techniques for fluid flow will then be given, namely the Finite Difference Method, The Finite Element Method and The Finite Volume Method. Further details of these methods will be given in subsequent Chapters.

We will then discuss briefly time discretisation and the order of accuracy of the numerical discretisation. This will be followed by an illustrative example to consolidate the concepts introduced in this Chapter.

3.1 Discretisation

3.1.1 Converting Derivatives to Discrete Expressions

The process of obtaining a numerical solution to a differential equation can be viewed in the same way as conducting a lab experiment. In a lab experiment, the physical quantity, flow velocity for example, is measured at discrete points in the domain of interest using a measurement device. A picture of the flow variation then can be constructed by connecting the measurement points allowing us to visualise the flow.

If we require the flow quantities between the measurement points, some interpolation technique can be used which may be linear or a higher order interpolation. This will depend on how far the points from each other, or how accurate we require these intermediate quantities.

In the same manner, numerical techniques convert the continuous differential equation to that of finding the solution at discrete points in space which we call grid points. A full picture of the flow then can be constructed from the solution at those points.

3.1.2 Spatial Discretisation

Let us assume that we want to represent a variation of a function θ , such as the one shown in Figure 3.1 using a polynomial in x of degree n .The function then can be expressed as:

$$\theta(x) = a_0 + a_1 x + a_2 x^2 + \ldots\ldots\ldots + a_n x^n \qquad (3.1)$$

We need then to employ a numerical method to find the coefficients $a_0, a_1 \ldots\ldots a_n$. Once these coefficients are determined, we are able to evaluate the function θ at any given value of x .

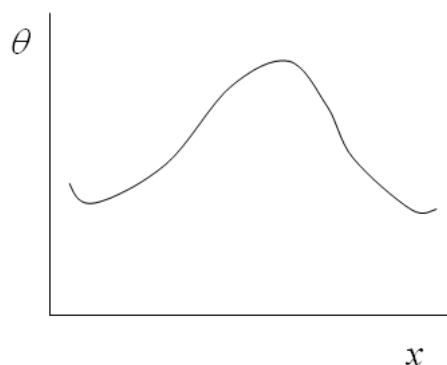



Figure 3.1: One dimensional function with x-coordinate

However, this procedure is inconvenient as every time we need the value of the function at a point, we need to substitute into the polynomial. The values of the coefficients themselves have no physical meaning and it is not easy to judge their validity by inspection.

A rather more convenient way is to employ the value of the dependent variable θ at the discrete points as the unknown which need to be determined numerically. The numerical methods to do that include providing a set of algebraic equations of these unknowns and finding an algorithm to solve those equations. This leads us to the concept of discretisation of the differential equation.

By focussing our attention on the values of the dependent variable at a set of grid points, we can replace the continuous information contained in the differential equation with discrete values. We thus can say that we have discretised the distribution of θ in the domain of interest.

The algebraic equations that involve the unknown values of the function θ at the grid points are derived from the governing differential equation by transferring the spatial derivatives to their discrete approximations as we shall see below.

We can thus view the discretised equations as algebraic relations connecting the approximations of the values of the dependent variable at the grid points. Since these algebraic equations are derived from the differential equation, they express the same physical information as that differential equation.

3.2 Discretisation Methods

We turn our attention now to the process of deriving the discretisation equations. The discretised form of a differential equation can be derived in many ways. We shall concentrate in this book on the most common methods used. The interested reader can refer to more advanced books for other techniques. These methods are explained in the following three subsections

3.2.1 The Finite Difference Method

The simplest procedure used to derive the discrete from of a differential equation, which we call here the Finite Difference equations consists of approximating the derivatives in the differential equation using a truncated Taylor series.

The Finite Difference Method is the simplest method to apply, particularly on uniform grids. However it requires high degree of mesh regularity. The mesh needs to be set up in a structured way where mesh points sould to be located at the intersection points of families of rectilinear curves.

We will give a brief description of the Finite Difference Method in this subsection. The method will be explained in more detail in Chapter 5.

Let us consider a one dimensional situation where the independent variable θ is a function of the space coordinate x as shown in Figure 3.1. We will discretise the spatial domain using equal space intervals of Δx and concentrate on three neighbouring points in the domain shown by the enlarged part of the domain in Figure 3.2. The three points are numbered arbitrarily as shown in the figure.

For grid point 2 in the middle between points 1 and 3, the Taylor series expansion gives the value of the field variable at point 1 as a function of the field variable and its derivatives at point 2 as follows:

$$\theta_1 = \theta_2 - \Delta x\left(\frac{d\theta}{dx}\right)_2 + \frac{1}{2}(\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right) - \ldots\ldots + \frac{1}{n}(\Delta x)^n\left(\frac{d^n\theta}{dx^n}\right) - \ldots\ldots \qquad (3.2)$$

Figure 3.2: Finite Difference discretisation

Similarly, the expansion gives the field variable at point 3 as:

$$\theta_3 = \theta_2 + \Delta x\left(\frac{d\theta}{dx}\right)_2 + \frac{1}{2}(\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right) + \ldots\ldots + \frac{1}{n}(\Delta x)^n\left(\frac{d^n\theta}{dx^n}\right) + \ldots\ldots \qquad (3.3)$$

By truncating the series in equation 3.2 after the 2nd term and rearranging, we obtain this expression for the first derivative at point 2:

$$\left(\frac{d\theta}{dx}\right)_2 = \frac{\theta_2 - \theta_1}{2\Delta x} \qquad (3.4)$$

This is called the first order backward difference approximation of the first derivative. We will elaborate on the order of approximation in Chapters 4 and 5.

Similarly, a first order forward difference can be obtained from equation 3.3.

By subtracting equation 3.3 from equation 3.2, a second order central difference approximation for the first derivative is obtained:

$$\left(\frac{d\theta}{dx}\right)_2 = \frac{\theta_3 - \theta_1}{2\Delta x} \qquad (3.5)$$

By adding equations 3.3 and 3.4, we obtain an expression for the second derivative at point 2:

$$\left(\frac{d^2\theta}{dx^2}\right)_2 = \frac{\theta_1 - 2\theta_2 + \theta_3}{(\Delta x)^2} \qquad (3.6)$$

If we substitute the expressions for the approximate derivatives into a differential equation, we obtain what is called the finite difference equation (See Section 3.3). Note that this leads to the transformation of the differential equation to an algebraic equation at point 2. Similar expressions can be obtained for all the points in the domain leading to a set of discrete algebraic equations.

Boundary conditions are applied by setting the known values at the end points. The system of algebraic equations can then be solved for the unknown quantities at the grid points.

We will look in more detail into how this can be used to numerically solve differential equations in later Chapters.

3.2.2 The Finite Element Method

The Finite Element Method is based on the so called 'Method of Weighted Residuals'. This is a powerful method for solving partial differential equations which was developed between 1940 and 1960, mainly for structural dynamics problems. This was extended later to the field of fluid flow.

This method has a distinct advantage over the Finite Difference method in the fact that it allows naturally for handling complex arbitrary geometries as it can be easily applied using irregular grids of various shapes. It also provides a set of functions that give the variation of the differential equations between grid points, whereas Finite Difference method provides information for the values at grid points only.

There is a wide body of literature describing the mathematical background of this method. We will resort here to simple description to convey the basic principles of the method to the student. Further details will be given in Chapter 6. Our focus in this book will be on the method of weighted residuals.

Assume that the differential equation at hand is of the general convection diffusion form as given in Equation 2.1. Taking the steady state version of this equation, to focus on the spatial discretisation, the equation becomes:

$$u\frac{\partial T}{\partial x} - \alpha\frac{\partial^2 T}{\partial x^2} = 0 \qquad (3.7)$$

For simplicity of presentation, this equation will be expressed in symbolic form as:

$$Q(T) = 0 \qquad (3.8)$$

Note that Equation 3.7 is given here for illustration purposes and the principles can be applied for any other differential equation.

If we assume that we are seeking an approximate solution T' using a trial function of some form. Then substituting this into the differential equation will not satisfy the equation. Thus a residual appears on the right hand side instead of zero. That is:

$$Q(T') = R \tag{3.9}$$

Because $Q(T')$ is an approximation, residual R does not vanish in most of the domain. The 'Weighted Residual Method' is based on the concept of introducing a weighting function W and then requiring that the integral of the weighted residual vanishes over the whole solution domain. That is:

$$\int_{\Omega} WQ(T')d\Omega = 0 \tag{3.10}$$

For the sake of argument, if we assume that the trial function is a polynomial with a number of unknown coefficients, then by selecting a succession of trial functions and integrating, a number of equations can be created which can be solved simultaneously to obtain the coefficients of the polynomial thus resulting in the solution.

This implies that the finite element method can be used to obtain analytic solutions to differential equations provided suitable trial and weighting functions can be found.

There are different techniques described in the literature used to define the trial and weighting functions such as the Sub-domain Method, the Collocation Method and the Least-Square Method. Detailed discussion of these methods is beyond the scope of this book. We will focus our attention here on the most popular method, namely the Galerkin Method, which will also be given further attention on Chapter 6.

Now we turn our attention on how to use this method to solve differential equations numerically. The first step is to assume local trial functions over the discretised domain. The solution domain is subdivided into non-overlapping cells, called elements. For example, in one dimension, these can be line segments between grid points. The trial functions are then interpolation functions which assume the shape of the variation of the variable between the grid points comprising the cell. The simplest of which are the linear shape functions that assume that the field variable has a linear variation between grid points. We then can express the solution as:

$$T'(x) = T_i N_i(x) \tag{3.11}$$

where N_i is the interpolation function at node i and T_i is the sought solution at that node. By choosing a series of weighting functions that have the value of W^i at node i and zero elsewhere for each node in the domain in turn, we can get an expression of the discrete weighted residual form by substituting into equation 3.10

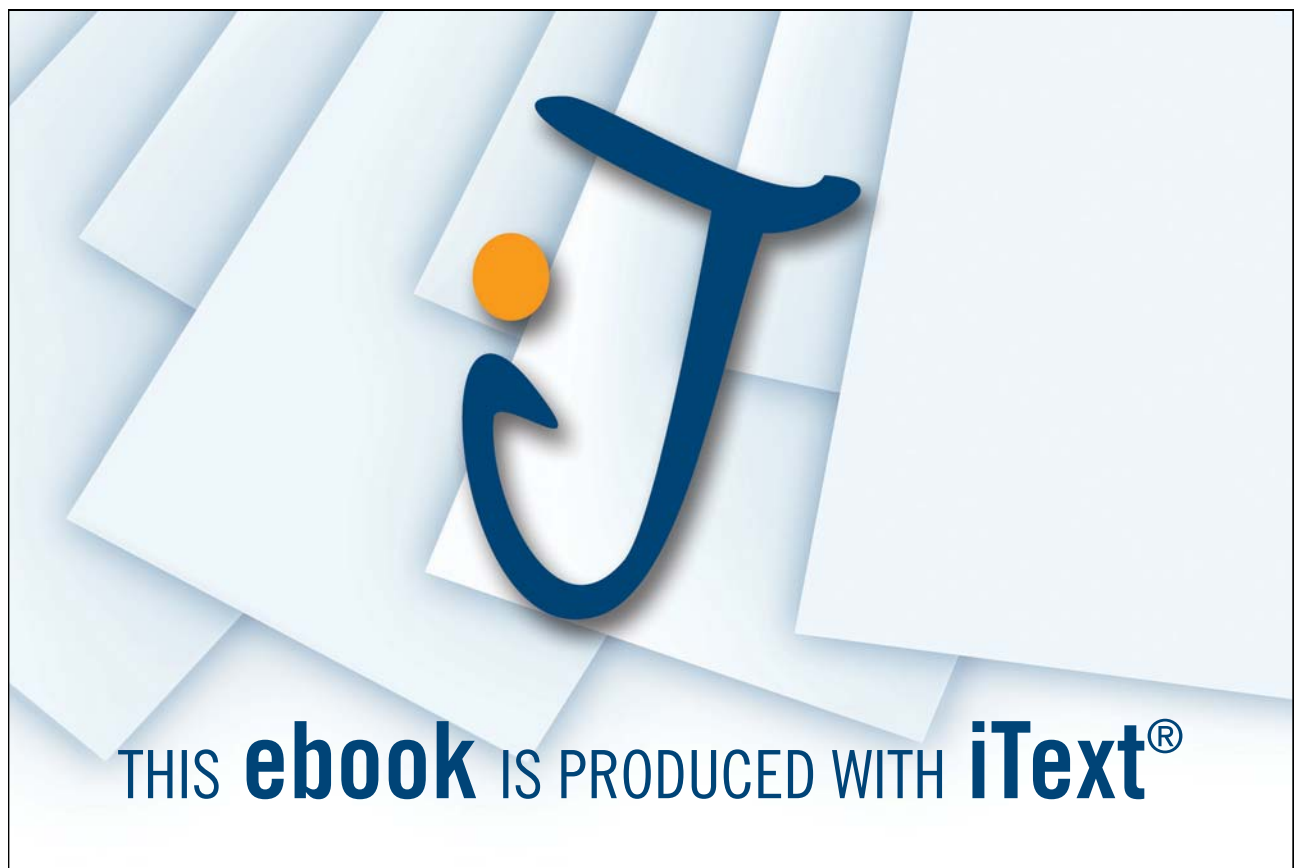
Substituting in equation, we obtain:

$$\int_{\Omega} W^i Q\big(T_i N_i(x)\big) d\Omega = 0 \tag{3.12}$$

Integrating at all the domain cells produces a system of algebraic equations of the form

$$K \cdot T_i = r \tag{3.13}$$

which can be solved for the coefficients T_i representing the field function at the nodal points. The matrix K is called the Jacobean or Mass matrix and the right hand side usually contains boundary conditions and source terms if there is any. We will discuss in more detail the formation of this system of equations in Chapter 6.

There are several possible choices for the approximation function and the weighting function. The most widely used choice is that weighting function is the same as the approximation function. This method is called the Galerkin Method.

3.2.3 The Finite Volume Method

This method was developed in the early 1970's. It can be viewed as a special case of the Weighted Residual Method described in the previous Section, where the weighting function takes the form:

$$W^i = 1 \tag{3.14}$$

For this, a number of weighted residual equations are generated by dividing the solution domain into sub-domains called 'control volumes' and setting the weighting function to be unity over the control volumes one at a time, and zero elsewhere. This implies that the residual over each volume must become zero.

Another way of deriving a finite volume discretisation is by starting from the integral form of the flow equations. Recall In Chapter 2 that we expressed the flow equations in their differential form. An alternative way of expressing the flow equations is the so called integral form.

For example, the continuity equation (Equation 2.5) can be expressed for a control volume Ω with a surface boundary Γ as:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho d\Omega + \oint_{\Gamma} \rho \vec{U} d\vec{\Gamma} = 0 \tag{3.15}$$

where $\vec{U} = \vec{u} + \vec{v} + \vec{w}$. This essentially states that the rate of accumulation of matter within domain Ω equals to the rate of the flux through its boundaries.

Similarly, integral formulations can be obtained for the momentum equations. For example, the integral form of the x-momentum equation takes the form:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho u d\Omega + \oint_{\Gamma} \left(\rho u \vec{U} + p - \tau\right) d\Gamma \tag{3.16}$$

where τ is the viscous flux given by:

$$\tau = \mu\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z}\right) \tag{3.17}$$

Similar integral forms can be written for the y and z momentum equations.

The finite volume formulation can start from this integral form. The fact that the variation of any quantity within a volume depends entirely on the surface values of the fluxes presents the basis of the Finite Volume formulation

The Finite Volume formulation starts by subdividing the solution domain into small volumes. We can then write the integral form of the conservation laws for each volume separately. The global conservation can be recovered by adding up the fluxes of the sub-volumes.

Let’s take for example the volume in Figure 3.3, which is divided to 4 sub-volumes

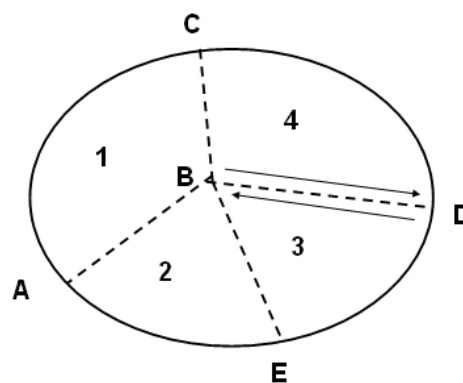



Figure 3.3: Finite Volume subdivisions

The flux through the internal subdivisions cancels out. For example the flux going through boundary BD of volume 3 is equal in magnitude and opposite in sign to the flux going through boundary DB of volume 4.

By working out the fluxes through all the boundaries on each sub-volume in terms of the field variable either at the volume centre point or at the vertices, a system of algebraic equations is constructed which can be solved for the unknown field variables. We will discuss this method in more detail in Chapter 7.

3.2.4 Temporal Discretisation

So far, we have discussed how the spatial domain (geometry) is transformed to a set of discrete nodes or volumes where a numerical scheme can be constructed converting the differential equation to its equivalent discrete algebraic form which can be solved for the unknown field variables. A large number of flow problems are unsteady in nature and the solution at a given point will vary with time. In this subsection, we will outline the discretisation of the time dimension.

Let's assume that we have discretised the spatial domain which has given us what we call semi-discrete form at time n:

$$\frac{\partial \phi}{\partial t} = f(\phi^n) \tag{3.18}$$

If we assume that the time is t at time step n and we wish to advance the solution to time step $n+1$ at time $t+\Delta t$. Note that time is a one-way coordinate and the evolution of the solution in time is obtained by marching in time for an initial given solution. The simplest form of time stepping is the explicit scheme.

In the explicit scheme, you predict the value of the field variable at the new time step based on its value only at the previous time step. So for Equation 3.18 we can write:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = f(\phi^n) \tag{3.19}$$

which can be rewritten as:

$$\phi^{n+1} = \phi^n + \Delta t.\left[f(\phi^n)\right] \tag{3.20}$$

Although this is a simple scheme to apply, usually, time steps that can be used while keeping the scheme stable are very small. As a result, other formulations are used such as implicit schemes and mixed schemes. Other time stepping schemes will be discussed in subsequent Chapters.

3.3 The Accuracy of the Discretisation Process

Since the process of converting the governing differential equations to a system of algebraic equations involves an approximation process, it is inevitable that the process will introduce a discretisation error.

If we focus our attention on the formulae used to work out the discrete form of the derivatives described in Section 3.2.1, we notice, as mentioned in that section, that the formula for the first derivative given is a one sided derivative. The order of accuracy of this formula can be inferred from the truncation error of the Taylor series used to derive the formula. We will demonstrate in Chapter 5 that the truncation error leads to a first order approximation. We will also show that the scheme including the two surrounding points is of second order accuracy.

In the general form, a finite difference representation of a derivative can be obtained by making a Taylor series expansion about a node at which the derivative is being evaluated. The evaluation of the remaining terms truncated from the series provides an approximation of the error for a given grid size and the more nodes we include in the calculation of the derivative, the higher the order of the scheme.

However, this lead to further complexity of the scheme and a balance has to be found between the accuracy of the scheme by reducing the truncation error and the accuracy of the scheme by keeping low order accuracy, but reducing the error by refining the grid.

3.4 Illustrative Example

We will use here the solution of a simple equation to illustrate the use of the Finite Difference Method to solve a differential equation in one dimension. The problem at hand is that of heat transfer in a cylindrical fin given in Figure 3.4, where both the base and the tip have fixed temperatures.

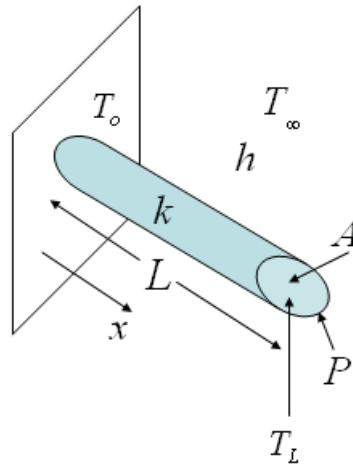



Figure 3.4: Heat transfer from a circular fin

If we define the non-dimensional temperature as:

$$\theta = \frac{T - T_\infty}{T_o - T_\infty}$$

and the axial coordinate as:

$$\zeta = \frac{x}{L}$$

Additionally, the characteristic convection conduction parameter is defined as:

$$\psi^2 = \frac{hP}{kA} L^2$$

where h is the convective heat transfer coefficient of the fluid around the fin, k is the conductivity of the fin material and P and A are the perimeter and cross sectional area of the fin respectively. It can be shown that the differential equation governing the temperature distribution along the fin is given by:

$$\frac{\partial^2 \theta}{\partial \zeta^2} - \psi^2 \theta = 0 \tag{3.21}$$

We are required to obtain a solution for this equation for the following conditions:

$$\psi^2 = 3$$

$$\theta_{base} = 1$$

$$\theta_{tip} = 0$$

Let us discretise the one dimensional domain using 5 cells of equal spacing resulting in 6 grid points as shown in Figure 3.5.

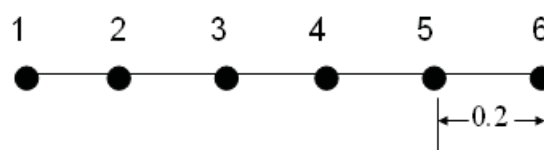



Figure 3.5: Grid used for the fin problem

Boundary conditions are given for points 1 and 6 as above. So for internal points, using equation 3.5 for the second derivative, a discrete form for each point can be written as follows:

$$\frac{\theta_{i-1} - 2\theta_i + \theta_{i+1}}{(0.2)^2} - 3\theta_i = 0$$

Rearranging:

$$25\theta_{i-1} - 53\theta_i + 25\theta_{i+1} = 0 \tag{3.22}$$

Thus if we use the boundary conditions for the end nodes and equation 3.22 for the inner nodes, we obtain the following set of algebraic equations for the discrete system:

$$\begin{aligned}
&\theta_1 = 1 \\
&25\theta_1 - 53\theta_2 + 25\theta_3 = 0 \\
&25\theta_2 - 53\theta_3 + 25\theta_4 = 0 \\
&25\theta_3 - 53\theta_4 + 25\theta_5 = 0 \\
&25\theta_4 - 53\theta_5 + 25\theta_6 = 0 \\
&\theta_6 = 0
\end{aligned} \tag{3.23}$$

This is a system of six equations containing six unknowns, which can be easily solved for the unknown temperatures. The solution for this systems and comparison with analytical solution will be left for the student as an exercise.

3.5 Closure

In this Chapter, we have introduced to the reader the concept of discretisation of a differential equation. We have also shown how a truncated Taylor series expansion can be used to express the continuous derivatives into their corresponding discrete form.

We then introduced three of the most common methods used for the spatial discretisation of the flow equations. These are the Finite Difference Method, the Finite Element Method and the Finite Volume method. This was followed by introducing the process of discretisation of the time derivative where explicit schemes were discussed.

A one-dimensional simple example was introduced showing how a differential equation describing a physical phenomenon is discretised over an equi-spaced grid using the Finite Difference method to illustrate the process.

In the following Chapters, we will elaborate in more details on the basic concepts introduced in this Chapter.

4 Properties of Numerical Schemes

In Chapter 2, we presented the mathematical models representing flow and we studied also the various simplifications to these models depending on the type of flow or the level of approximation required. In Chapter 3, we presented the basic principles of discretising these mathematical models by converting them to a set of algebraic equations at discrete nodes. These equations can be solved using digital computers.

In this Chapter, we will discuss the basic rules that govern the numerical schemes to make them valid methods for obtaining the desired solutions. The three basic properties are consistency, stability and convergence.

4.1 Basic definitions

The concepts of consistency, convergence and stability will be illustrated by way of an example using a simple representative model. Let's take the constant coefficient convection equation below:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \qquad (4.1)$$

where u is the unknown velocity field in the x-direction and a is the constant convection speed. Note that we have previously discussed this problem and its solution in Chapter 2.

If we consider the following initial conditions at time $t = 0$:

$$\begin{aligned} u &= 0 & x &\le 2 \\ &= -4 + x & 2 &< x \le 4 \\ &= 6 - x & 4 &< x < 6 \\ &= 0 & x &\ge 6 \end{aligned} \qquad (4.2)$$

This initial solution is shown in Figure 4.1 in red colour.

Let's use the 2nd order difference scheme (Equation 3.5) to discretise the equation in space with explicit advancing in time from time step *n* to *n+1*. We obtain the following discrete form for Equation 4.1:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{2\Delta x}\left(u_{i+1}^n - u_{i-1}^n\right) \qquad (4.3)$$

This equation can be written as:

$$u_i^{n+1} = u_i^n - \frac{a\Delta t}{2\Delta x}\left(u_{i+1}^n - u_{i-1}^n\right) \tag{4.4}$$

The term $\sigma = \frac{a\Delta t}{\Delta x}$ is usually called the Courant number. This equation can be advanced in time to obtain a solution at time *n+1* for a given grid with a choice of Δt given that a is constant and Δx is fixed by the space discretisation. This solution is shown after 1, 2 and 3 time steps together with the exact solution in Figure 4.1.

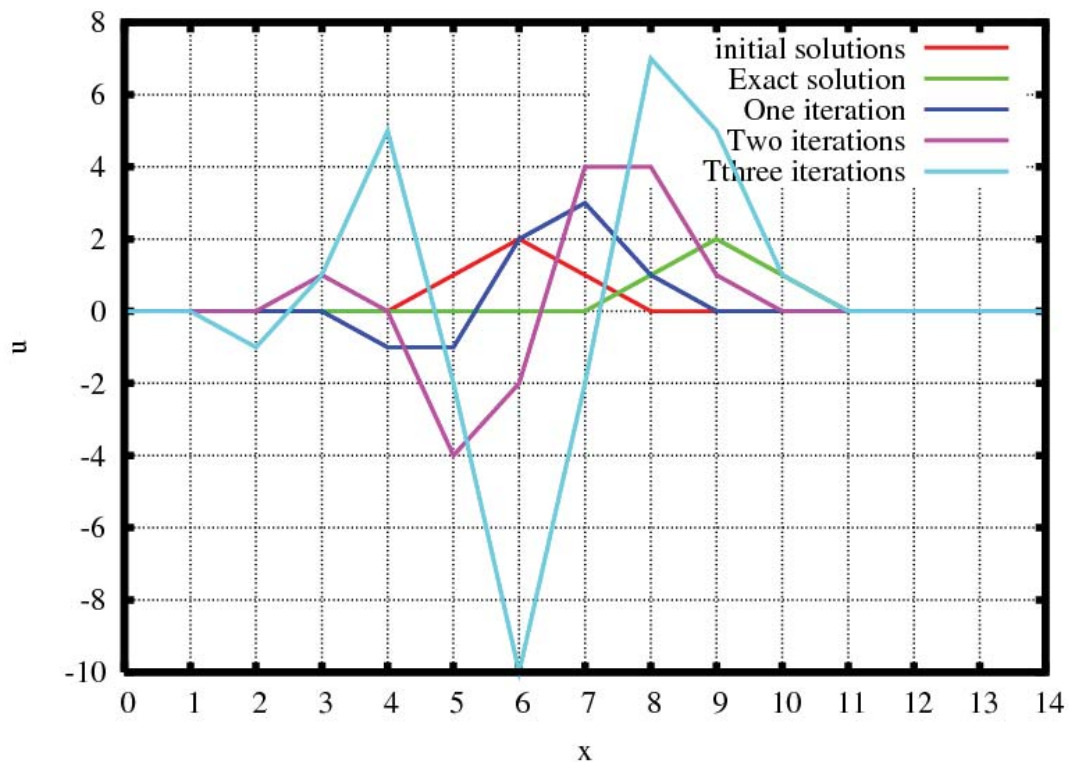



Figure 4-1: Solution using central difference scheme

The values of σ which was used here is 2.0. It is noticed that the solutions is unstable as it leads to increasing error with time. Tests show that this instability happens whatever value is given for σ indicating that the scheme is always unstable.

Let's now use the first order backward difference scheme in space given by Equation 3.4 instead of the 2nd order central difference scheme above. The backward difference scheme here is referred to as an upwind scheme as the information used is from the point upstream. This gives the following discretised equation with the same explicit time discretisation:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -\frac{a}{\Delta x}\left(u_i^n - u_{i-1}^n\right) \tag{4.5}$$

which can be rearranged to give:

$$u_i^{n+1} = u_i^n - \sigma\left(u_i^n - u_{i-1}^n\right) \tag{4.6}$$

The solution in this case for $\sigma = 0.5$ is shown in Figure 4.2. It can be seen that the solution is stable as it progresses in time. It reaches a solution which looks like the exact solution. However, is not as accurate. There is some spreading and reduction in amplitude as if there is some diffusion. We will discuss the reason for this in Chapter 5. However, let's now focus on the stability of the shceme.

If we try the solution with $\sigma = 1.5$ we get an unstable solution as in Figure 4.3. This is typical of a conditionally stable scheme as the stability depends on the value of σ.

Figure 4-2: Solution using upwind scheme with Courant number 0.5

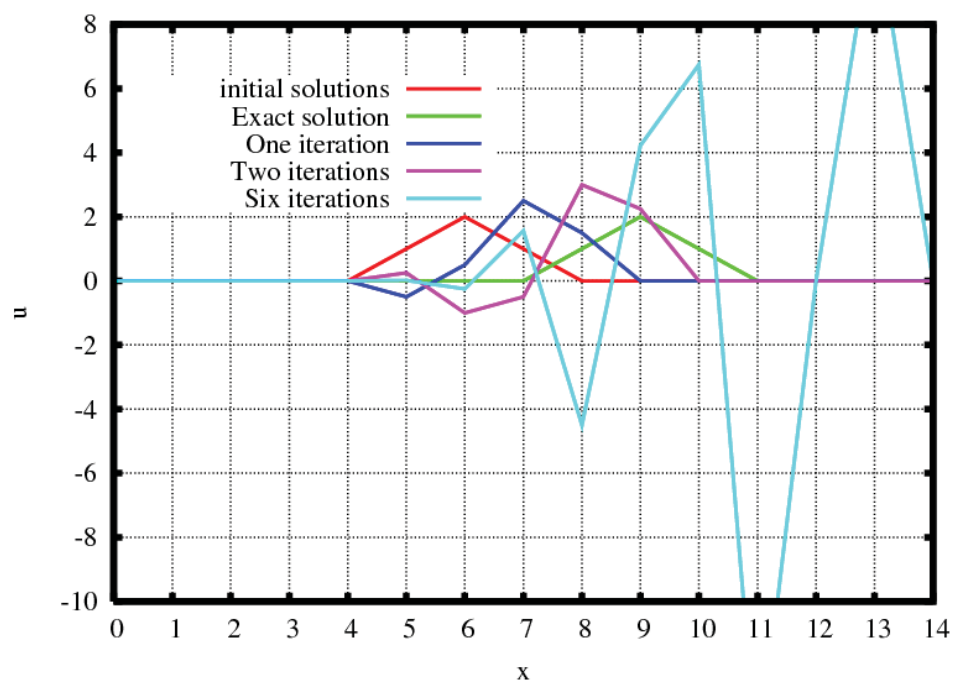



Figure 4-3: Solution using upwind space scheme with Courant number 1.5

The above discussion illustrated several important points. The two most obvious ones are the stability and convergence. A less obvious property which will also be discussed is consistency. We will discuss each of these properties in turn in the following sections.

4.2 Consistency

For a numerical discretisation to be consistent, it should tend to the differential equation which it is related to as Δx and Δt tend to zero. Consistency is a necessary condition for the approximate solution but not sufficient. As we will demonstrate below, the discretisation in equation 4.5 satisfies the consistency condition, however, convergence does not occur for $\sigma > 1/2$.

One way of checking the consistency of a numerical discretisation is to use a Taylor's series expansion for all the terms with the higher order terms retained. The expansion is then substituted into the discretised equation and the resulting equation is checked for consistency (i.e. if it tends to the original differential equation as Δx and Δt tend to zero). To illustrate this by an example, we will use the discretisation given in Equation 4.5.

$$u_i^{n+1} = u_i^n + \Delta t\left(\frac{du}{dt}\right)_i^n + \frac{\Delta t^2}{2}\left(\frac{d^2u}{dt^2}\right)_i^n + \text{.........} \tag{4.7}$$

$$u_{i-1}^n = u_i^n - \Delta x\left(\frac{du}{dx}\right)_i^n + \frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i^n - \text{.........} \tag{4.8}$$

Substituting these approximations in equation 4.5

$$\frac{u_i^n + \Delta t\left(\frac{du}{dt}\right)_i^n + \frac{(\Delta t)^2}{2} + \text{.....} - u_i^n}{\Delta t} = -\frac{a}{\Delta x}\left(u_i^n - u_i^n + \Delta x\left(\frac{du}{dx}\right)_i^n - \frac{(\Delta x)^2}{2}\left(\frac{d^2u}{dx^2}\right)_1^n + \text{....}\right)$$

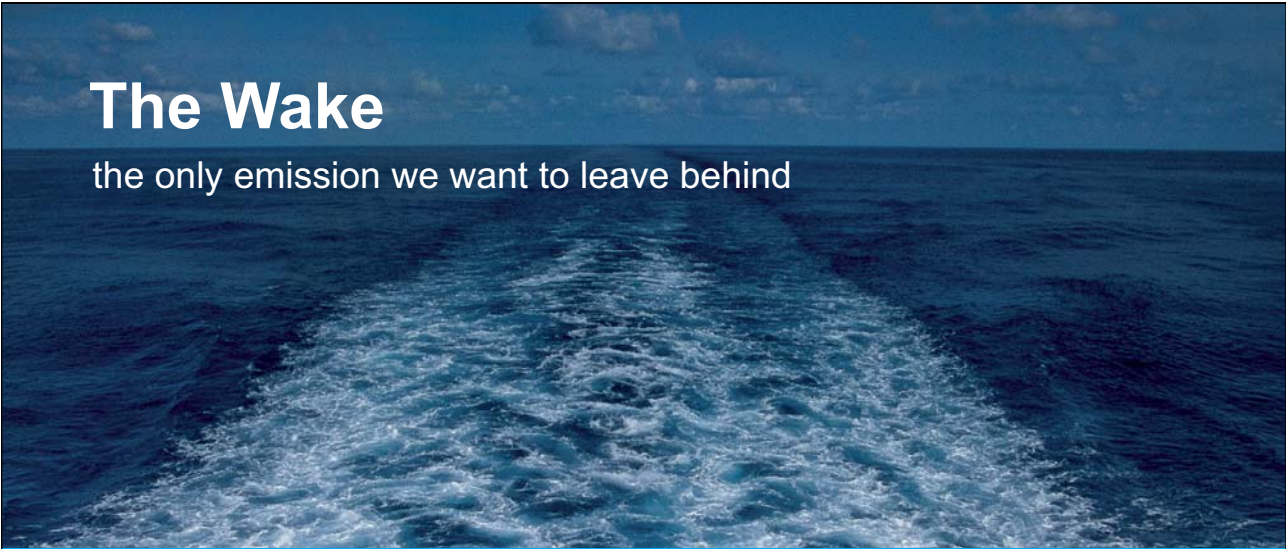
Rearranging leads to:

$$\left(\frac{du}{dt}\right)_i^n + a\left(\frac{du}{dx}\right)_i^n = \frac{a\Delta x}{2}\left(\frac{d^2u}{dx^2}\right)_i^n - \frac{\Delta t}{2}\left(\frac{d^2u}{dt^2}\right) + o(\Delta x^2, \Delta t^2) + \text{.........} \tag{4.9}$$

Clearly, the right hand side of Equation 4.9 tends to zero as Δx and Δt tend to zero leaving the left hand side, which is equivalent to the original differential equation. This indicates that the forward in time, upwind scheme of Equation 4.5 is consistent with the original differential equation.

It might appear from the above simple example that consistency can be taken for granted. However, attempts to construct algorithms that are both accurate and stable might lead to inconsistencies in discretisation. Hence a check is always needed.

4.3 Stability

Stability is the tendency of any perturbations in the solution of the discretised system of equations to decay. So if these perturbations grow without bound, the scheme is unstable. We have demonstrated an unstable scheme in Section 4.1, for all conditions. That was the forward time central difference scheme applied to the advection equation with constant coefficients.

On the other hand, we have also shown in the same section an example of a conditionally stable scheme in the form of the forward in time, upwind difference scheme for the same equation.

The concept of stability is concerned with the growth or decay of errors introduced at any stage of the computation. The errors are not referred to those introduced by the incorrect logic of the numerical scheme or the solution procedure, but those which occur because the computer cannot give answers to an infinite number of decimal places.

In practice, the computations made on the computer are carried out to a finite number of significant figures depending on the hardware or software or both on the particular computer. This introduces what is known as round-off errors at every time step or iteration of the computations.

A particular method is stable if the cumulative effect of the round-off errors produced in the application of the numerical algorithm is negligible, or does not build up continuously at every time step.

We were able to determine stability properties of the two simple schemes mentioned above by way of a simple example. However, it is generally difficult to do that for every scheme. Hence, formalised techniques have been developed to study the stability of numerical schemes.

The three most common methods of stability analysis are the method of equivalent differential equations, the matrix method and the von-Neumann method. The three methods are based on predicting whether there will be a growth in error between the true solution of the numerical algorithm and the actually computed solution.

The detailed description of these methods is beyond the scope of this book and the interested reader is referred to Hirsch (1988).

An alternative interpretation of stability analysis is to assume that the initial conditions are represented by a Fourier series. Each harmonic in the Fourier series will decay or grow depending on the discretised equation which typically leads to a specific expression for the growth or decay factor for each mode. If one of the modes can grow without bounds, the discretised equation will have an unstable solution. For more details, see Fletcher (1991).

4.4 Convergence

Convergence means that the numerical solution should approach the exact solution of the differential equation at any point in the domain when Δx and Δt approach zero. This means that as the mesh is refined and the time step is reduced, the solution should be approaching the exact solution.

In practical situations, a proof that the resulting solution is converging to the exact solution is difficult as the exact solution is not known and the aim of the numerical approach is to get a solution that is not possible analytically. For this reason, convergence in that context is very difficult to prove.

In many numerical procedures the concept of convergence is used differently. For example, in iterative procedures, the residual is monitored which is the difference between the solution at a particular iteration and the next iteration for steady problems. It is required then that this residual approaches zero for the solution to converge. This, in fact, means that the solution to the system of discretised algebraic equations has been obtained. This does not necessarily mean that the solution is converged to the exact solution.

Similarly, for procedures with pseudo time stepping, where steady state solutions are obtained by advancing a system of equations in pseudo time until the time derivative approaches zero, which means the solution has approached its required steady-state solution, the term convergence is also used. In this case again, this means that the solution to the discretised equations have been achieved, but it does not mean necessarily that the solution has converged to the exact solution.

The alternative way of assessing convergence, which is a common practice, is what is called grid independence or mesh convergence. In this case, successive solutions are obtained on finer and finer girds until the solution variation from one mesh to the next mesh approaches a prescribed tolerance. In this case, the solution is assumed to be approaching the exact solution. But care should be exercised here as other factors and approximation in the model or associated sub-models might influence the solution to a degree or another and it might lead to a false interpretation of the results.

5 The Finite Difference Method

In Chapter three we presented a brief overview of the most popular methods used for the discretisation of differential equations. The finite difference method was introduced and it was mentioned that it is the simplest one conceptually. However, this method is difficult to apply when we are encountered with complex geometries. For this reason, this method is of limited use for practical applications and only a very small number of engineering codes rely on this method.

However, the simplicity of the method allows us to explore the properties of various numerical discretisations and compare their degree of accuracy. It also allows us to have a better grasp of numerical procedures. Additionally, for solution procedures which require higher order derivatives or high order of accuracy, this method can be better suited than other methods despite the limitation of mesh regularity.

5.1 Basics

In Chapter 2 we have used the Taylor's series expansion to transform the terms in a differential equation to their discrete counterpart at grid points. We will start with this to build an understanding of the accuracy of the various formulations.

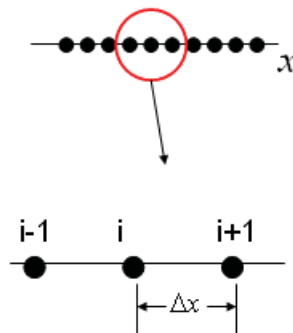



Figure 5-1: Finite difference stencil

Let's start with the Finite Difference stencil shown in Figure 5.1. A Taylor's series expansion around point i in terms point $i-1$ gives:

$$\theta_{i-1} = \theta_i - \Delta x\left(\frac{d\theta}{dx}\right)_i + \frac{1}{2}(\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right)_i - \frac{1}{3}(\Delta x)^3\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \qquad (5.1)$$

Rearranging Equation 5.1 gives:

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_i - \theta_{i-1}}{\Delta x} + \frac{1}{2}\Delta x\left(\frac{d^2\theta}{dx^2}\right)_i - \frac{1}{3}(\Delta x)^2\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \qquad (5.2)$$

If we truncate the right hand side after the 2nd term, we obtain the expression:

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_i - \theta_{i-1}}{\Delta x} \qquad (5.3)$$

The expression in Equation 5.3 is called the backward difference and it is first order accurate. The order of accuracy is related to the power of Δx in the first truncated term of the Taylor's series, which is one in this case.

Similarly, a first order accurate forward difference can be obtained by a Taylor's series expansion in terms of point $i+1$ as follows:

$$\theta_{i+1} = \theta_i + \Delta x\left(\frac{d\theta}{dx}\right)_i + \frac{1}{2}(\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right)_i + \frac{1}{3}(\Delta x)^3\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \qquad (5.4)$$

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_{i+1} - \theta_i}{\Delta x} + \frac{1}{2}\Delta x\left(\frac{d^2\theta}{dx^2}\right)_i + \frac{1}{3}(\Delta x)^2\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \qquad (5.5)$$

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_{i+1} - \theta_i}{\Delta x} \qquad (5.6)$$

If we add Equations 5.2 and 5.4 and divide both sides by 2 gives:

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta x} + 0 + \frac{1}{3}(\Delta x)^2\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \qquad (5.7)$$

Truncating equation 5.7 after the first term gives:

$$\left(\frac{d\theta}{dx}\right)_i = \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta x} \qquad (5.8)$$

Equation 5.8 is a central difference scheme of 2nd order accuracy because the first truncated term is in the order of $(\Delta x)^2$. That is the power of Δx for the first truncated term is two. A geometrical

interpretation of the three difference schemes is shown in Figure 5.2. It can be seen that the central difference approximation provides a better representation of the slope of the curve at the point of interest which is the first derivative.

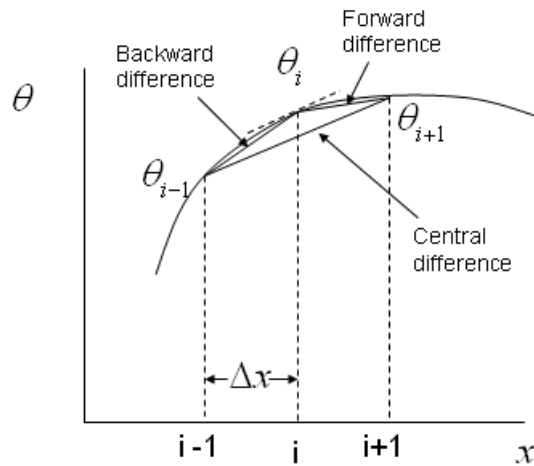



Figure 5-2: Geometric interpretation of difference formulae

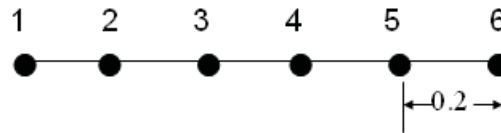
Example

In this example, we will revisit the fin example presented in Section 3.2. The objective is to show how the finite difference gradient boundary condition can be applied. The objective is to solve the equation:

$$\frac{\partial^2\theta}{\partial\zeta^2} - \psi^2\theta = 0 \tag{5.9}$$

with the boundary condition $\frac{\partial\theta}{\partial\zeta} = 0$ at $x = L$, that is at $\zeta = 1$. This is a more realistic boundary condition than prescribed temperature. It indicates that the heat flux at the tip of the fin is zero, indicating that temperature gradient is zero.

Solution

Let's use the same discretisation of 6 grid points as in Section 3.2, and use the central difference scheme for all internal nodes (node 2-5) as before giving the discrete equation of

$$25\theta_{i-1} - 53\theta_i + 25\theta_{i+1} = 0$$

At point 1, the temperature is specified by the base temperature and thus we do not need to elaborate further as we have:

$$\theta_1 = 1$$

However, for point 6, there is a problem if we attempt to use the central difference scheme as there is no grid point to the right of point 6. Remember that we need to apply the gradient boundary condition at this point. We could be tempted to use a backward difference formula (Equation 5.3). At the first instance, this might seem a good idea. Let's examine that.

Applying Equation 5.3 to point 6 and imposing the condition of $\frac{\partial \theta}{\partial \zeta} = 0$ gives:

$$\frac{\theta_6 - \theta_5}{0.2} = 0$$

This leads to the equation $\theta_6 = \theta_5$

This can be used with the other set of equations to solve the problem, however, an error is found near that point which propagates to the rest of the domain. This error results from the fact that the discretisation at point 6 is first order accurate which is not consistent with the second order accurate scheme for the internal nodes. Figure 5.3 shows the solution compared to the analytic solution for this case.

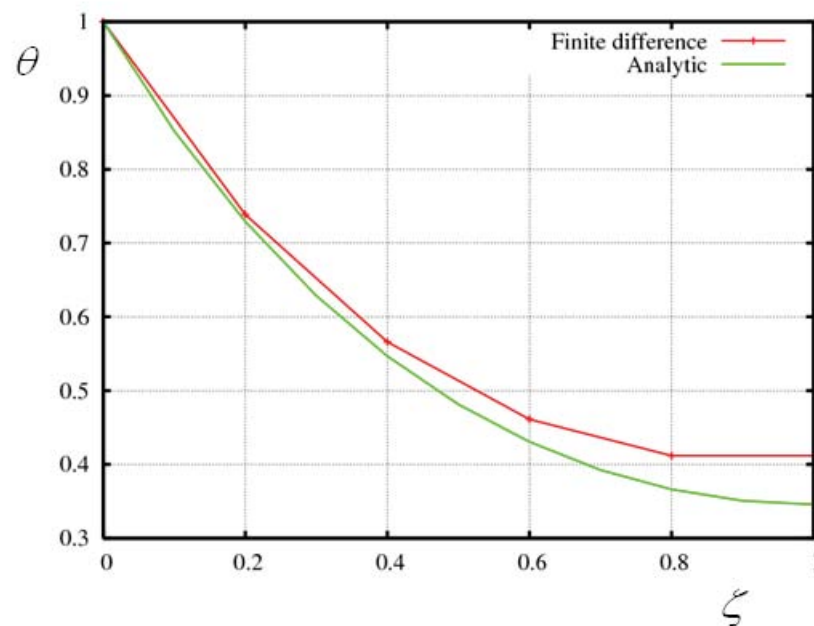



Figure 5-3 Solution with first order discretisation of boundary conditions

overcome this problem, the boundary conditions equation needs to be discretised with the same order of accuracy as the internal points. To enable this, we add an imaginary point after nodes 6, lets say node 7, and we work out the value of the field variable at this node in terms of the values at nodes 5 and 6. This then allows us to formulate a central difference equation. Thus, with reference to Figure 5.4:

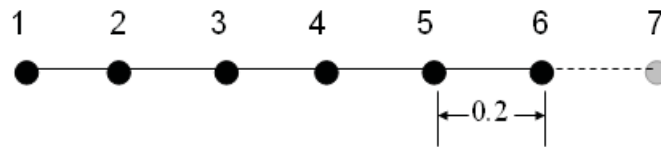



Figure 5-4 Applying gradient boundary condition at point 6

The central difference formula at point 6 is then:

$$25\theta_5 - 53\theta_6 + 25\theta_7 = 0$$

which can be solved for θ_7 to give:

$$\theta_7 = \frac{53\theta_6 - 25\theta_5}{25} \tag{5.10}$$

The second order gradient boundary condition for point 6 is then:

$$\frac{\partial\theta}{\partial\zeta} = \frac{\theta_7 - \theta_5}{0.4} = 0 \tag{5.11}$$

Substituting from 5.10 into 5.11:

$$\frac{\dfrac{53\theta_6 - 25\theta_5}{25} - \theta_5}{0.4} = 0$$

which can be rearranged to give:

$$53\theta_6 - 50\theta_5 = 0$$

Which is clearly different from that obtained using the first order backward formula above. Using this equation together with the other equations for the rest of the nodes leads to the solution shown in Figure 5.5 which is in a much better agreement with the analytic solution.

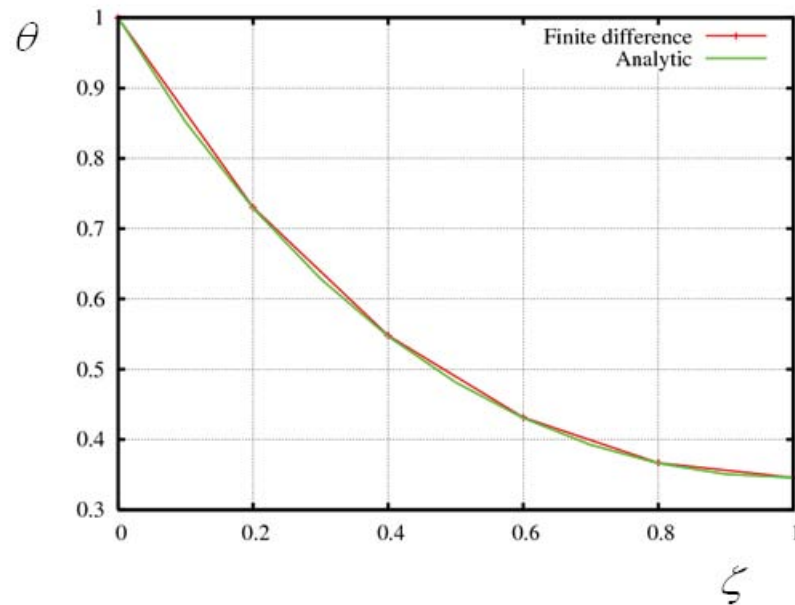



Figure 5-5 Solutions with second order discretisation of boundary conditions

5.2 Other difference formulae

We have seen that difference formulae for the first derivative can be formulated using one or two adjacent points. However, finite difference formulae for the first derivative can be formulated using any number of adjacent points with the order of approximation increasing with the number of points.

In a particular numerical scheme, a balance need to be found between the order of accuracy and the number of grid points involved in the computation. This will dictate the computational memory and effort both need to be kept to a minimum for a given overall solution accuracy.

It is usually difficult to devise rules that govern the optimum combination of discretisation accuracy and number of grid points as these vary from problem to problem and only experience with a large number of cases can produce guidance in this respect.

Let's now examine the process of finding formulae involving more grid points for a given accuracy. Consider the stencil shown in Figure 5.6.

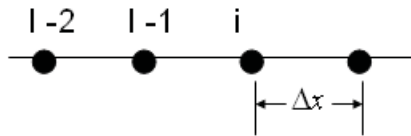



Figure 5-6: Stencil for second order backward difference formula

If we are required to formulate a second order backward difference formula, we use the following procedure:

$$\frac{d\theta}{dx} = \frac{a\theta_i + b\theta_{i-1} + c\theta_{i-2}}{\Delta x} + O\left[(\Delta x)^2\right] \tag{5.12}$$

where *O* in the last term on the right hand side means order-of. We will then use Taylor's Series expansions to create a system of equations that can be solved for the coefficients a, b and c. Hence

$$\theta_{i-1} = \theta_i - \Delta x\left(\frac{d\theta}{dx}\right)_i + \frac{1}{2}(\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right)_i - \frac{1}{3}(\Delta x)^3\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \tag{5.13}$$

$$\theta_{i-2} = \theta_i - 2\Delta x\left(\frac{d\theta}{dx}\right)_i + (2\Delta x)^2\left(\frac{d^2\theta}{dx^2}\right)_i - \frac{1}{3}(2\Delta x)^3\left(\frac{d^3\theta}{dx^3}\right)_i + \ldots\ldots \tag{5.14}$$

Multiplying Equation 5.13 by b and Equation 5.14 by c and adding $a\theta_i$ we get:

$$a\theta_i + b\theta_{i-1} + c\theta_{i-2} = \\ (a+b+c)\theta_i + \Delta x(2c+b)\left(\frac{d\theta}{dx}\right)_i + \frac{(\Delta x)^2}{2}(4c+b)\left(\frac{d^2\theta}{dx^2}\right)_i - O\left[(2\Delta x)^3\right] \tag{5.15}$$

Comparing this to Equation 5.12, we find that only the coefficients of the term containing the first derivative will not vanish. This gives the following three Equations:

$$a + b + c = 0$$

$$-(2c + b) = 1$$

$$4c + b = 0$$

Solving simultaneously gives:

$$a = \frac{3}{2}, \quad b = -2, \quad c = \frac{1}{2}$$

Thus the second order difference formula becomes:

$$\frac{d\theta}{dx} = \frac{1.5\theta_i - 2\theta_{i-1} + 0.5\theta_{i-2}}{\Delta x} \tag{5.16}$$

The above procedure with undetermined coefficients can by systematically used to obtain finite difference formulae for all derivatives at any required degree of accuracy. As an exercise for the student, try to derive a second order forward formula using a three point stencil to give the following:

$$\frac{d\theta}{dx} = \frac{-1.5\theta_i + 2\theta_{i+1} - 0.5\theta_{i+2}}{\Delta x} \tag{5.17}$$

5.3 Multi-dimensional Finite difference formulae

One of the advantages of the finite difference method is its straight forward extension to multi-dimensions. The way that partial derivatives of a function of several variables can be discretised using the same methods of the previous sections for each variable and for each coordinate direction.

To illustrate this lets consider the thermal conduction problem in two dimensions, which is also know as Laplace's equations.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{5.18}$$

Our objective is to discretise this equation for a two dimensional domain. Let's first consider internal nodes. Figure 5.7 shows part of the two dimensional finite difference grid with a stencil centred on point m, n. The index m is incremented in the x-direction, while the index n is incremented in the y direction

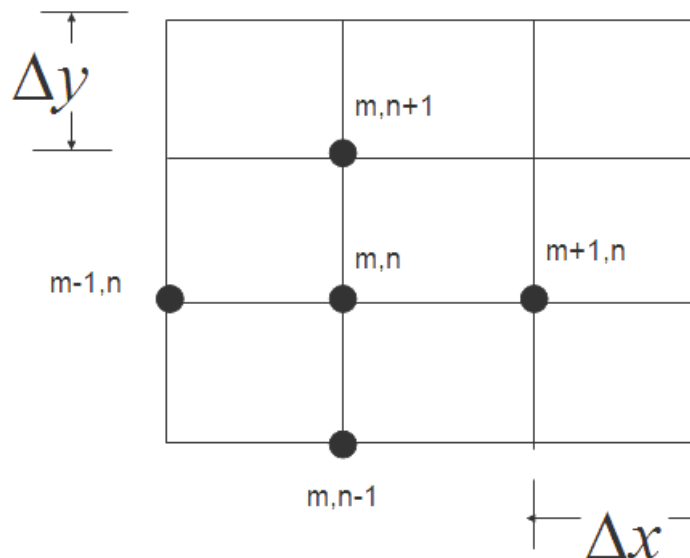



Figure 5-7:Two-dimensional Finite Difference Stencil

If we apply the second order difference formula of Equation 3.6 for both x and y direction separately, we obtain the following discrete form for node m,n:

$$\frac{T_{m+1,n} - 2T_{m,n} + T_{m-1,n}}{(\Delta x)^2} + \frac{T_{m,n+1} - 2T_{m,n} + T_{m,n-1}}{(\Delta y)^2} = 0 \qquad (5.19)$$

If we assume that the grid spacing is equal in both x and y directions, that is $\Delta x = \Delta y$, then Equation 5.19 reduces to:

$$T_{m+1,n} + T_{m-1,n} + T_{m,n+1} + T_{m,n-1} - 4T_{m,n} = 0 \qquad (5.20)$$

A similar equation can be obtained for each internal grid node. For boundary nodes, the discretisation will depend on the type of boundary conditions applied. If a given temperature boundary conditions is to be applies, then the equation for that node is deleted and replaced by the given value for the temperature.

If a flux boundary condition is to be applied, then a procedure similar to that explained in Section 5.2 is applied. In this case a fictious node is added and the value of the temperature at that node is computed as a function of the internal nodes. The same central difference scheme then can be used at boundary nodes.

The resulting discretisation leads to a number of equations, which equals the number of unknown temperatures. These equations can then be solved simultaneously for the unknown nodal temperatures. We will discuss in more detail solution methods to systems of algebraic equations in Chapter 8.

The case described here illustrates the simplicity of extension of the finite difference method to two dimensions. Extension to three-dimensions is equally simple and follows the same logic.

5.4 Non-Uniform meshes

For geometries with boundaries that do no coincide with the coordinate systems, Finite difference method still can be used. In this case, there are two possible approaches. The first is to generate a regular grid as shown in Figure 5.8. All internal nodes can be treated in the usual way as described in the previous section.

For nodes on the boundary, some grid cells will not be complete. In this case, a special procedure is required for the nodes surrounding those cells. There are several ways to approximate the discretised equations at those nodes. These might include shifted stencils that use the inner nodes or fictious nodes outside the domain. In either case, the fact that cell is not complete need to be taken into account.

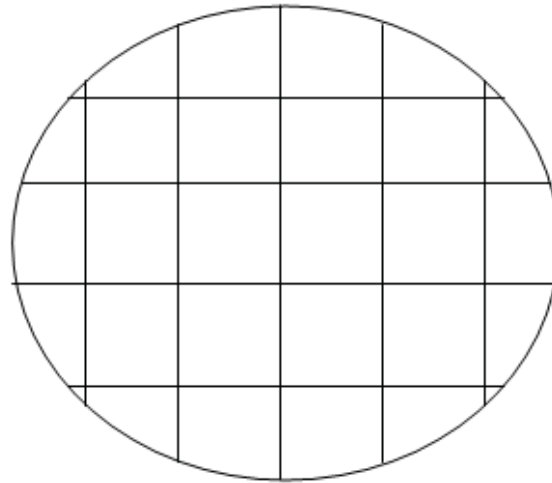



Figure 5-8: Cartesian grid for a circular domain

The other alternative is to generate a body fitted grid (see Chapter 10). In this case, the grid is what is termed the physical plane is mapped onto a regular grid in a computational plane (See Figure 5.9). The differential equations are then transformed onto the computational plane using the same mapping.

Once this is done, all the standard difference formulae discussed before can be applied in the computational domain. Once the discrete equations are constructed, they can be solved and the results mapped back to the physical domain, or the discrete equations are mapped back to the physical domain and solved.

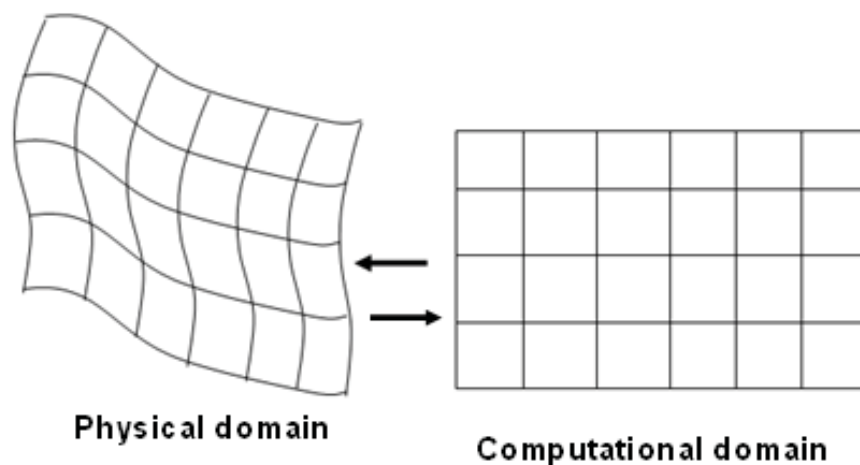



Figure 5-9 mapping a body fitted grid to a regular grid

6 The Finite Element Method

The Finite Element Method was briefly introduced in Section 3.2.2. We then outlined the history of the method, its advantages. We also outlined the theoretical background and the framework by which the Galerkin weighted residual method is used to discretise differential equations.

In this Chapter, we will present the method in more detail. There is a wide volume of literature discussing the theoretical background of the Finite Element Method and its application to engineering problems. If you are interested in more detail, refer to one of the reference listed in the back of this Chapter.

The concept of the Finite Element Method can be traced back to the technique used in stress calculations whereby a structure was divided into small sub-structures of various shapes called 'elements'. The structure is then re-assembled after each element has been analysed.

The technique was developed further in what is now known as the Finite Element Method between 1940 and 1960, mainly in the field of structural dynamics. The technique was then expanded to solve field problems in the 1960's (See Zeinkiewicz 1977).

Nowadays, the Finite Element Method has been put in an engineering rigorous framework with precise mathematical conditions for existence, convergence and error bounds. In this book, we will not concentrate on the mathematical derivation of the method, but rather on its application for the discretisation of differential equations.

6.1 Basics

As discussed in previous Chapters, a numerical model for fluid flow starts with a physical model of the problem. We could choose a model of the full Navier-Stokes equations or any of the approximation levels. We then require solving the mathematical model over a given physical domain with some boundary conditions.

The first step is to discretise the spatial domain into non-overlapping elements or sub-regions. The Finite Element Method allows a variety of element shapes, for example, triangles, quadrilaterals in two dimensions and tetrahedral, hexahedral, pentahedral, and prisms in three dimensions. Each element is formed by the connection of a certain number of nodes, with the number of nodes in an element depending on the type of the element (Figure 6.1).

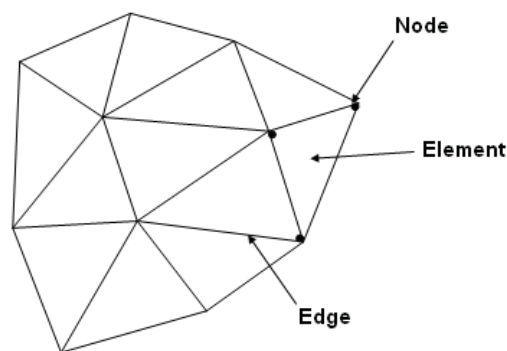



Figure 6-1: Typical two-dimensional Finite Element mesh

The number of nodes in each element does not depend only on the number of corner points in the element, but also on the type of the element interpolation function as we will explain in the next section.

Once a mesh is generated, we choose the type of interpolation function that represents the variation of the field variable over the element. A clear distinction can be seen here from the Finite Difference Method. In the Finite Difference Method, we were only interested in the values of the field variable at grid nodes, and no information was required for the behaviour between the nodes. We may have implicitly assumed that it is linear, but we did not have to do that.

The next stage is to determine the matrix equations that express the properties of the individual element by forming a left hand side matrix and a load vector. A typical left hand side matrix and a load vector for a one dimensional element may look like:

$$[K]_e = A\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{6.1}$$

$$\{f\}_e = \begin{Bmatrix} q_i \\ q_j \end{Bmatrix} \tag{6.2}$$

where A contains some geometric and/or physical parameters of the element.

The next stage is to assemble the element equations to obtain a system of simultaneous equations that can be solved for the unknown field variables at the mesh nodes. The final system of equations will be represented in matrix notation as:

$$[K]\{U\} = \{f\} \tag{6.3}$$

The vector $\{U\}$ is the vector of the unknown field variables at the nodes.

In the next section, we will illustrate those steps using a simple example. But before we do that, we will present more information about the element shape functions and the Finite Element discretisation process of differential equations.

6.2 Elements and Shape Functions

The Finite Element Method involves both the discretisation of the computational domain and the discretisation of the differential equations. In this process, the variables are represented in a piece-wise manner over the domain. By dividing the solution domain into elements, and approximating the solution over these elements using a suitable known function, a relationship between the elements and the differential equation is established.

The functions used to represent the variation of the solution within each element are called shape functions, or interpolation functions or basis functions. Typically, polynomial functions are used because they can be easily integrated or differentiated. The accuracy of the results can be improved by increasing the order of the polynomial used.

6.2.1 One-dimensional elements

These are the simplest elements and their discussion will help illustrate the basic principles. The simplest element has a piece-wise linear interpolation function and contains two nodes. Let's consider the function shown in Figure 6.2. If the x-coordinate is discretised to a set of elements each containing two nodal points, and the function is approximated using linear variation between each two nodes, then the variation for a typical element is shown in Figure 6.3.

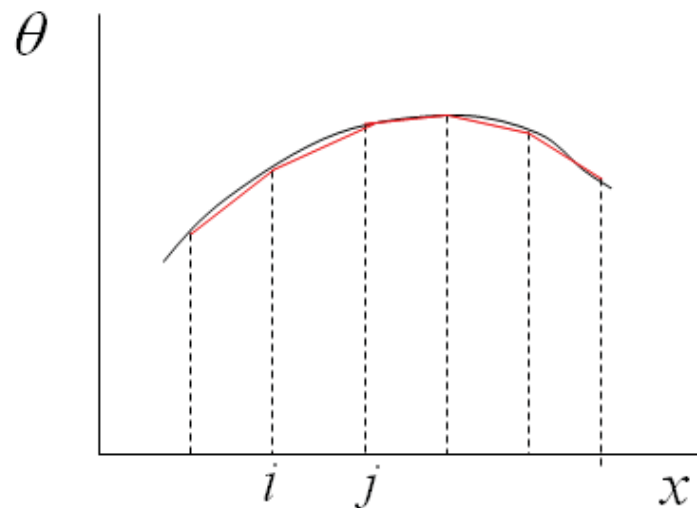



Figure 6-2: Linear piece-wise representation in one-dimension

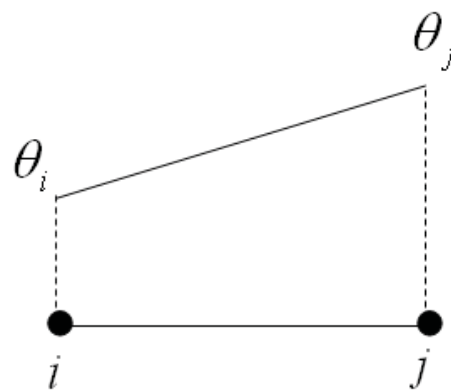



Figure 6-3: Linear variation over one element

The linear variation over the element of Figure 6.3 can be represented using the summation of two shape functions N_i and N_j at nodes i and j respectively. Each shape function will have a maximum of unity and the corresponding point and varies linearly to zero at the other point as shown in Figure 6.4.

The shape functions can be expressed mathematically as:

$$N_i = \frac{x_j - x}{x_j - x_i} \tag{6.4}$$

$$N_j = \frac{x - x_i}{x_j - x_i} \tag{6.5}$$

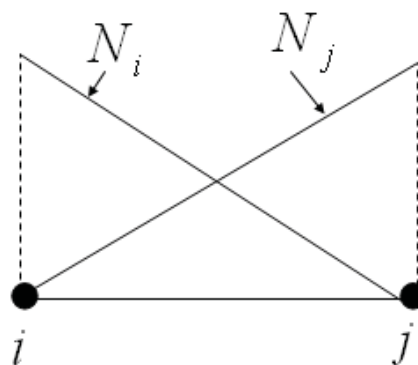



Figure 6-4: Linear shape functions in one dimension

The field function is then represented over the element using the shape function as:

$$\theta_e = N_i\theta_i + N_j\theta_j \tag{6.6}$$

This is essentially the same as the linear variation in Figure 6.3. We can use the expression in Equation 6.6 to get the first derivative of the field variable within the element. That is:

$$\frac{d\theta_e}{dx} = \frac{dN_i}{dx}\theta_i + \frac{dN_j}{dx}\theta_j \tag{6.7}$$

$$\frac{d\theta_e}{dx} = \frac{-1}{x_j - x_i}\theta_i + \frac{1}{x_j - x_i}\theta_j \tag{6.8}$$

Noting that $x_j - x_i$ is the length of the element. Let's give this length the symbol l . Then Equation 6.8 can be written in the following compact matrix notation:

$$\frac{d\theta_e}{dx} = \frac{1}{l}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}\theta_i \\ \theta_j\end{bmatrix} \tag{6.9}$$

We can observe from this also that the first derivative of the field variable is constant over the element. This indicates that the first derivative of the function over the domain will be stepwise constant over the entire domain indicating that it is not a continuous function.

Higher order shape functions can be obtained by using more nodes within the element. For example a quadratic shape function can be obtained by using three nodes in the element as shown in Figure 6.5. Using a quadratic polynomial, the shape functions can be derived to give (with the length of the element $l = x_k - x_i$):

$$N_i = 1 - \frac{3x}{l} + \frac{2x^2}{l^2} \tag{6.10}$$

$$N_j = \frac{4x}{l} + \frac{4x^2}{l^2} \tag{6.11}$$

$$N_k = -\frac{x}{l} + \frac{2x^2}{l^2} \tag{6.12}$$

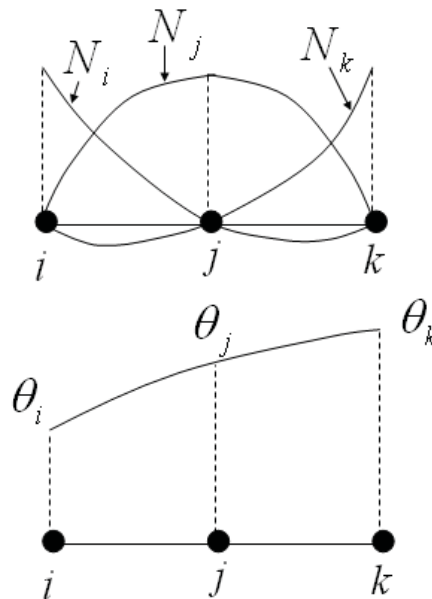



Figure 6-5 Quadratic element and shape functions

And the field variable is represented over the element by:

$$\theta_e = N_i\theta_i + N_j\theta_j + N_k\theta_k \tag{6.13}$$

Exercise:

As an exercise, the student is required to derive a matrix expression for $\frac{d\theta_e}{dx}$ using the quadratic element shape function similar to that derived for the linear element in Equation 6.9.

6.2.2 Two-dimensional triangular elements

The most popular element for arbitrary two dimensional geometries is the triangular element. This is mainly because triangular meshes are relatively easier to generate and to control their quality. We will discuss methods if generating triangular grids in Chapter 9. In this section, we will present the shape functions for these elements.

A two dimensional linear element is shown in Figure 6.6. We can represent the variation of the field function on this element using a linear polynomial follows:

$$\theta_e(x, y) = a + bx + cy \tag{6.14}$$

If the field functions at the three grid point are labelled θ_1, θ_2 and θ_3, then the three coefficients a, b and c can be determined as follows:

$$\theta_1 = a + bx_1 + cy_1 \tag{6.15}$$

$$\theta_2 = a + bx_2 + cy_2 \tag{6.16}$$

$$\theta_3 = a + bx_3 + cy_3 \tag{6.17}$$

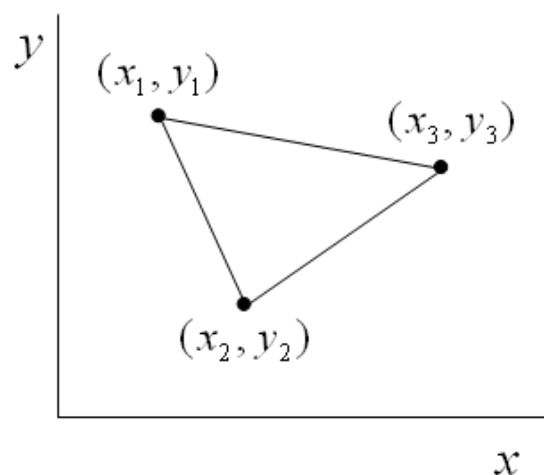



Figure 6-6: Two dimensional linear triangular element.

Equations 6.15-6.17 can be solved simultaneously to get the shape function coefficients in terms of the nodal coordinates to give:

$$a = \frac{1}{2A}\left[(x_2y_3 - x_3y_2)\theta_1 + (x_3y_1 - x_1y_3)\theta_2 + (x_1y_2 - x_2y_1)\theta_3\right] \tag{6.18}$$

$$b = \frac{1}{2A}\left[(y_2 - y_3)\theta_1 + (y_3 - y_1)\theta_2 + (y_1 - y_3)\theta_3\right] \tag{6.19}$$

$$c = \frac{1}{2A}\left[(x_3 - x_2)\theta_1 + (x_1 - x_3)\theta_2 + (x_2 - x_1)\theta_3\right] \tag{6.20}$$

where A is the area of the triangle given by:

$$A = \frac{1}{2}\left[(x_1y_2 - x_2y_1) + (x_3y_1 - x_1y_3) + (x_2y_3 - x_3y_2)\right] \tag{6.21}$$

Substituting the values of the three coefficients from Equations 6.18-6.20 into Equations 6.15-6.17 and gathering coefficients we obtain the following expression for Equation 6.14:

$$\theta_e(x, y) = N_1\theta_1 + N_2\theta_2 + N_3\theta_3 \tag{6.22}$$

where the shape functions are given by:

$$N_1 = \frac{1}{2A}\left[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right] \tag{6.23}$$

$$N_2 = \frac{1}{2A}\left[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right] \tag{6.24}$$

$$N_3 = \frac{1}{2A}\left[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right] \tag{6.25}$$

Equation 6.22 can be written using the following matrix notation:

$$\theta_e(x, y) = \begin{bmatrix} N_1 & N_2 & N_3 \end{bmatrix}\begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} \tag{6.25}$$

The shape functions have some properties that are worth noticing. The first is that if we evaluate the shape function at its corresponding node, for example if we evaluate N_1 at node 1, we obtain $2A/2A = 1$ and if we evaluate N_1 at nodes 2 or 3, we get zero. Additionally, at any point in the triangle:

$$N_1 + N_2 + N_3 = 1 \tag{6.26}$$

The gradient of the field variable over the element can be calculated as follows:

$$\begin{aligned}\frac{\partial \theta}{\partial x} &= \frac{\partial N_1}{\partial x}\theta_1 + \frac{\partial N_2}{\partial x}\theta_2 + \frac{\partial N_3}{\partial x}\theta_3 \\ \frac{\partial \theta}{\partial y} &= \frac{\partial N_1}{\partial y}\theta_1 + \frac{\partial N_2}{\partial y}\theta_2 + \frac{\partial N_3}{\partial y}\theta_3\end{aligned} \tag{6.27}$$

which can be written as follows after getting the derivatives of the shape functions:

$$\begin{aligned}\frac{\partial \theta}{\partial x} &= \frac{y_2 - y_3}{2A}\theta_1 + \frac{y_3 - y_1}{2A}\theta_2 + \frac{y_1 - y_2}{2A}\theta_3 \\ \frac{\partial \theta}{\partial y} &= \frac{x_3 - x_2}{2A}\theta_1 + \frac{x_1 - x_3}{2A}\theta_2 + \frac{x_3 - x_2}{2A}\theta_3\end{aligned} \tag{6.28}$$

Equation 6.28 can be written using the following compact matrix notation:

$$\begin{bmatrix} \dfrac{\partial \theta}{\partial x} \\ \\ \dfrac{\partial \theta}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} (y_2 - y_3) & (y_3 - y_1) & (y_1 - y_2) \\ \\ (x_3 - x_2) & (x_1 - x_3) & (x_3 - x_2) \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} \tag{6.29}$$

It should be noted, as in the linear one dimensional element, that the first derivative of the field function is constant within the triangular two dimensional element.

Higher order triangular elements can be obtained by placing more nodes within the element and using higher order polynomials to obtain the shape functions. As most CFD methods use the linear elements, we will not analyse higher order elements in detail and the interested reader can refer to one of the books in the reference list of this book.

It is sufficient in this book to show some of these high order elements. Figure 6.7 shows the nodes on a quadratic and a cubic triangular element.

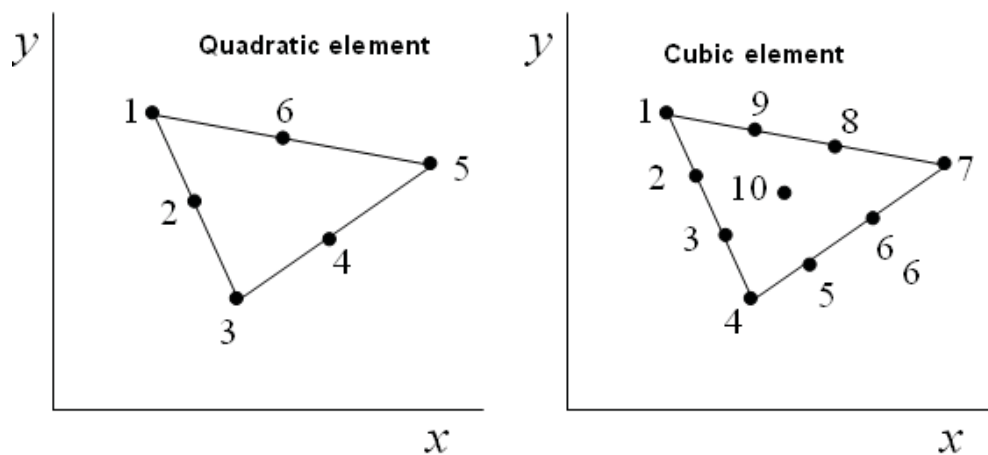



Figure 6-7: Examples of high order triangular elements

6.2.3 Two dimensional quadrilateral elements

Bilinear quadrilateral elements have four nodes located at the vertices as shown in Figure 6.8. A quadrilateral Finite Element grid may look like a Finite Difference grid. But in the Finite Difference mesh, the grid need to be orthogonal, that is all grid lines intersect at right angles, whereas in the Finite Element mesh, this restriction is removed and each element can have a unique shape.

In addition, In the Finite Difference mesh, each grid point has to have the same number of neighbours. For example, in two dimensional grids, each grid point should be surrounded by four points, while in Finite Element Method; a point can have an arbitrary number of neighbours. Unstructured quadrilateral grids with arbitrary number of neighbours for each node can be generated using paving techniques that will be discussed in Chapter 9.

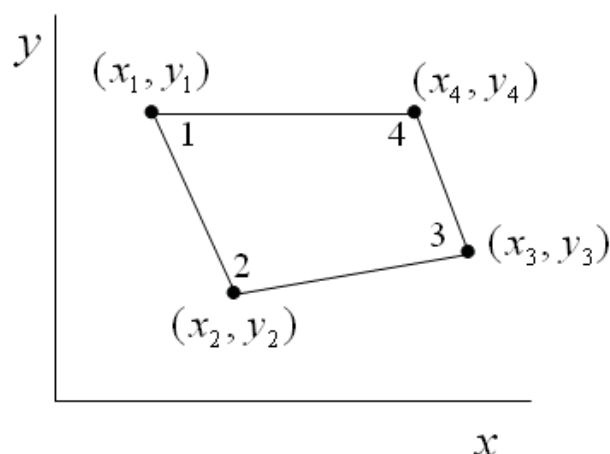



Figure 6-1: Bilinear quadrilateral element

The field variable within a quadrilateral element can be expressed using the polynomial:

$$\theta_e(x, y) = a + bx + cy + d\ xy \tag{6.30}$$

where a, b, c and d are coefficients that need to be determined to define the shape functions as we did with the triangular element. Since we would like to deal with elements of general shapes, it is usually more convenient to map the element using geometric transformation to a master space where the elements in the local coordinates ζ and η are of regular shape and vary in coordinates as shown in Figure 6.9. Then the shape functions are defined on the master element, or isoperimetric element. The inverse transformation function can be used to transform the discretised system to the actual physical space.

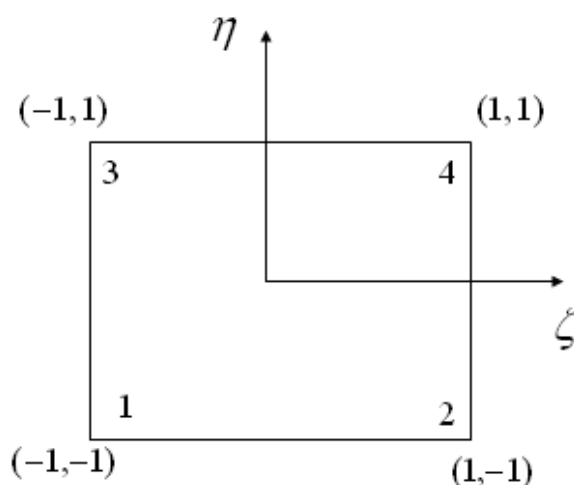



Figure 6-9: Isoparametric element mapping for a quadrilateral element

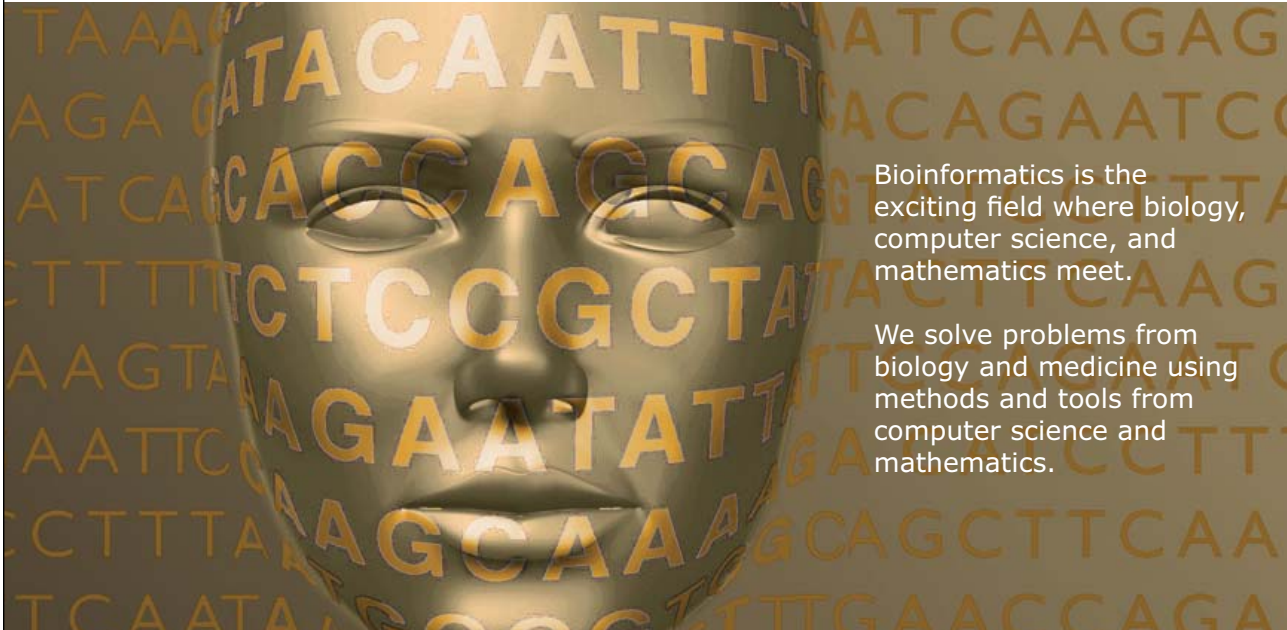
We seek to define shape functions at the four nodes such that:

$$\theta_e(x,y) = N_1\theta_1 + N_2\theta_2 + N_3\theta_3 + N_4\theta_4 \tag{6.31}$$

We will not go into the derivation of these shape functions and the interested reader is referred to Zeinkeiwichz (1977). However, we will state she shape functions in terms of the local coordinates. These take the form:

$$N_1 = \frac{1}{4}(1-\zeta)(1-\eta) \tag{6.32}$$

$$N_2 = \frac{1}{4}(1+\zeta)(1-\eta) \tag{6.33}$$

$$N_3 = \frac{1}{4}(1+\zeta)(1+\eta) \tag{6.34}$$

$$N_4 = \frac{1}{4}(1-\zeta)(1+\eta) \tag{6.35}$$

To obtain a transformation between the physical coordinates and the local or isoparametric coordinates, we can use the expression in Equation 6.31 as follows:

$$x = N_1 x_1 + N_2 x_2 + N_3 x_3 + N_4 x_4 \tag{6.36}$$

$$y = N_1 y_1 + N_2 y_2 + N_3 y_3 + N_4 y_4 \tag{6.37}$$

Equations 6.36 and 6.37 can be written in the following compact notation:

$$x = \sum_{i=1}^{4} N_i x_i \tag{6.38}$$

$$y = \sum_{i=1}^{4} N_i y_i \tag{6.39}$$

From both equations, we get:

$$\frac{\partial x}{\partial \zeta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \zeta} x_i, \qquad \frac{\partial x}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} x_i \tag{6.40}$$

$$\frac{\partial y}{\partial \zeta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \zeta} y_i, \qquad \frac{\partial y}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} y_i \tag{6.41}$$

where the derivatives in of the shape functions in the summation are:

$$\frac{\partial N_1}{\partial \zeta} = \frac{1}{4}(1-\eta), \qquad \frac{\partial N_1}{\partial \eta} = -\frac{1}{4}(1-\zeta) \tag{6.42}$$

$$\frac{\partial N_2}{\partial \zeta} = \frac{1}{4}(1-\eta), \qquad \frac{\partial N_2}{\partial \eta} = -\frac{1}{4}(1+\zeta) \tag{6.43}$$

$$\frac{\partial N_3}{\partial \zeta} = \frac{1}{4}(1+\eta), \qquad \frac{\partial N_3}{\partial \eta} = \frac{1}{4}(1+\zeta) \tag{6.44}$$

$$\frac{\partial N_4}{\partial \zeta} = -\frac{1}{4}(1+\eta), \qquad \frac{\partial N_4}{\partial \eta} = \frac{1}{4}(1-\zeta) \tag{6.45}$$

And using the chain rule, we can write:

$$\frac{\partial N_i}{\partial \zeta} = \frac{\partial N_i}{\partial x}\frac{dx}{d\zeta}, \qquad \frac{\partial N_i}{\partial \zeta} = \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \zeta} \tag{6.46}$$

$$\frac{\partial N_i}{\partial \eta} = \frac{\partial N_i}{\partial x}\frac{dx}{d\eta}, \qquad \frac{\partial N_i}{\partial \eta} = \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \eta} \tag{6.47}$$

Equations 6.46 and 6.47 can be written in the compact matrix notation as:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \zeta} \\ \frac{\partial N_i}{\partial \mu} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \tag{6.48}$$

This can be expressed as:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \zeta} \\ \frac{\partial N_i}{\partial \mu} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \tag{6.49}$$

And using the expressions in Equations 6.40 and 6.41 we get

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} \frac{\partial N_i}{\partial \zeta} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \zeta} y_i \\ \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} x_i & \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix} \tag{6.50}$$

Thus the derivatives of the shape functions with respect to the physical space can be given by:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \zeta} \\ \frac{\partial N_i}{\partial \mu} \end{Bmatrix} \tag{6.51}$$

The result in Equation 6.51 is an important expression for the finite element formulation of differential equations because it allows working out the derivatives of the field variable as follows:

$$\begin{aligned}\frac{\partial \theta}{\partial x} &= \frac{\partial N_1}{\partial x}\theta_1 + \frac{\partial N_2}{\partial x}\theta_2 + \frac{\partial N_3}{\partial x}\theta_3 + \frac{\partial N_4}{\partial x}\theta_4 \\ \frac{\partial \theta}{\partial y} &= \frac{\partial N_1}{\partial y}\theta_1 + \frac{\partial N_2}{\partial y}\theta_2 + \frac{\partial N_3}{\partial y}\theta_3 + \frac{\partial N_4}{\partial y}\theta_4\end{aligned} \tag{6.52}$$

Higher order quadrilateral elements can be obtained in a similar manner to higher order triangular elements by placing more nodes in the element and using higher order polynomials to represent the shape functions. Figure 6.10 shows for example an eight nodded quadrilateral element.

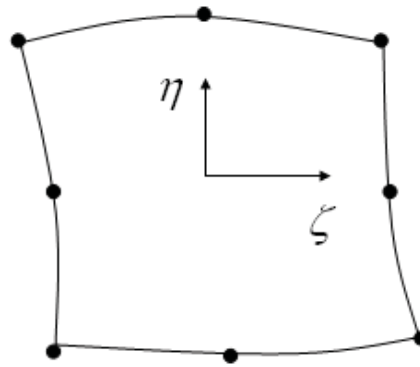



Figure 6-10: Eight nodded quadrilateral element

6.2.4 Three-Dimensional elements

Three dimensional elements' shape functions can be derived in the same manner as one and two dimensional elements. The additional spatial coordinate allows for the possibility of more type of elements. The three most popular three dimensional linear elements are tetrahedral, hexahedral and prismatic elements shown in Figure 6.11.

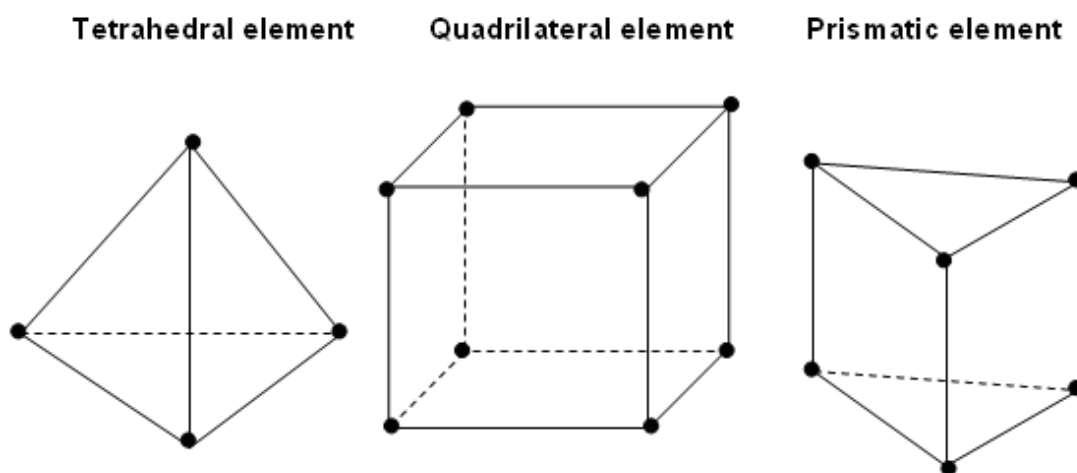



Figure 6-11: Three-dimensional elements

Because of the larger number of nodes in three dimensional elements, the amount of data to required to establish the shape functions become significantly greater than two-dimensional elements. Consequently, the amount of computational time and memory required become significantly greater.

This means that care should be exercised when generating three-dimensional grids to get the most appropriate element for a particular application which produces the least possible computational effort.

Detailed analysis of the shape functions will not be presented in this introductory book and the interested reader is referred to Zeinkeiwickz (1977).

6.3 Weighted Residual Method

It was explained in Chapter three that there are several methods to derive a Finite Element formulation for differential equations. It was also mentioned that the method of weighted residuals, particularly the Galerkin method is the most popular one.

We are not interested in the theoretical derivation of this method. Our interest here is mainly on how to apply this method to transform a given differential equation into its equivalent discrete form over a Finite Element grid. We will illustrate this by way of an example for a one-dimensional problem that we addressed in Chapters 3 and 5. That is the fin problem of Section 3.2.

The choice of this example serves at least two purposes. The first is the illustration of the Finite Element discretisation using the Galerkin method. The second, if we use the same regular grid that was used for the Finite Difference Discretisation, we can compare the two discretisations.

We want to discretise the equation:

$$\frac{\partial^2\theta}{\partial\zeta^2}-\psi^2\theta=0 \qquad (6.53)$$

Using the 5 element grid shown in Figure 6.12 with the boundary conditions: $\theta_1 = 1$ and $\theta_6 = 0$

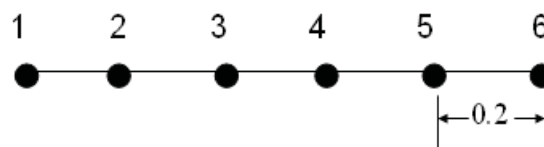



Figure 6-12: Finite Element grid for equation 6.53

We can start the discretisation using the shape functions of the linear one-dimensional element of Equations 6.5 and 6.6. However to follow a more general procedure that can be used in two and three-dimensional isoprametric elements, we will express the one-dimensional shape functions using isoparametric one-dimensional element.

If we pick any of the elements in Figure 6.53 and map it to a local coordinate system as shown in Figure 6.54, the origin is defined at the mid point of the element, then substituting $x_i = -1$ and $x_j = 1$ in Equations 6.5 and 6.6, we get the following shape functions:

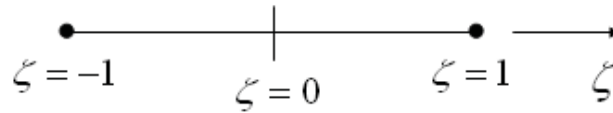



Figure 6-3: Isoparametric one-dimensional linear element

$$N_i = \frac{\zeta - 1}{-1-1} = \frac{1}{2}(1-\zeta) \qquad (6.54)$$

$$N_i = \frac{\zeta - (-1)}{-1-(-1)} = \frac{1}{2}(1+\zeta) \qquad (6.55)$$

Thus the approximate field function expressed over the element is given by:

$$\theta_e = N_i\theta_i + N_j\theta_j \tag{6.56}$$

And the gradient over the element is given by:

$$\frac{d\theta_e}{d\zeta} = \frac{dN_i}{d\zeta}\theta_i + \frac{dN_j}{d\zeta}\theta_j = -\frac{1}{\zeta}\theta_i + \frac{1}{\zeta}\theta_j \tag{6.57}$$

The next stage is to use the Galerkin formulation outlined in Section 3.2.2, by which we substitute the approximation of the field variable from Equation 6.56 into the differential equation, multiply this by a weighting function which is of the same form as the field shape function and require that the integral of this weighted residual to equal zero over the solution domain. Hence:

$$\int_\Omega N_k\left(\frac{d^2\theta_e}{d\zeta^2} - \psi^2\theta_e\right)d\zeta = 0 \tag{6.58}$$

The subscript k indicates the nodes in the domain. This means that by using the shape functions at all nodes one at a time, we obtain a set of equations which equals the number of nodes. Since the first derivative of the linear shape function (Equation 6.56) is constant, then the second derivative is zero. To be able to represent the second derivative, we need to obtain the so called weak formulation, by which the first term of Equation 6.58 is integrated by parts. So for the first node of the above element:

$$\int_\Omega N_i\left(\frac{d^2\theta_e}{d\zeta^2}\right)d\zeta = \vec{n}\left[N_i\frac{d\theta}{d\zeta}\right]_0^{\zeta_e} - \int_0^{\zeta_e}\frac{dN_i}{d\zeta}\frac{d\theta}{d\zeta}d\zeta \tag{6.59}$$

where $\vec{n}$ is the outward normal to the boundary which equals 1, with appropriate sign. The end result when the equations are assembled together is that this term cancels out for all internal nodes and will be either 1 or -1 at the boundary nodes.

Substituting into Equation 6.58, and replacing the field function and its derivative by their approximate representations from Equations 6.56 and 6.57, we get:

$$\begin{aligned}\int_\Omega N_i\left(\frac{d^2\theta_e}{d\zeta^2} - \psi^2\theta_e\right)d\zeta = \vec{n}\left[N_i\frac{d\theta}{d\zeta}\right]_0^{\zeta_e} &- \int_0^{\zeta_e}\left(\frac{dN_i}{d\zeta}\frac{dN_i}{d\zeta}\theta_i + \frac{dN_i}{d\zeta}\frac{dN_j}{d\zeta}\theta_j\right)d\zeta \\ &- \int_0^{\zeta_e}\psi^2\left(N_iN_i\theta_i + N_iN_j\theta_j\right)d\zeta\end{aligned} \tag{6.60}$$

Substituting the values of the shape function and its derivative in terms of ζ and integrating, we obtain an equation for node i which can be written in the following matrix notation; with the outward normal at node 1 is 1.

$$\frac{1}{\zeta_e}\begin{bmatrix}1 & -1\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \frac{\psi^2\zeta_e}{6}\begin{bmatrix}2 & 1\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \begin{Bmatrix}\dfrac{d\theta}{d\zeta} \\ 0\end{Bmatrix} \tag{6.61}$$

If we weight the equation with N_j and repeat the integration, we obtain the following equation for the second node of element 1, where the outward normal here is -1 because it is in the opposite direction:

$$\frac{1}{\zeta_e}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \frac{\psi^2\zeta_e}{6}\begin{bmatrix}2 & 1\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \begin{Bmatrix}0 \\ -\dfrac{d\theta}{d\zeta}\end{Bmatrix} \tag{6.62}$$

We can now assemble the contributions of nodes 1 and 2 of the first element in one matrix representing the element to give:

$$\frac{1}{\zeta_e}\begin{bmatrix}1 & -1 \\ -1 & 1\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \frac{\psi^2\zeta_e}{6}\begin{bmatrix}2 & 1 \\ 1 & 2\end{bmatrix}\begin{Bmatrix}\theta_i \\ \theta_j\end{Bmatrix} + \begin{Bmatrix}\dfrac{d\theta}{d\zeta} \\ -\dfrac{d\theta}{d\zeta}\end{Bmatrix} \tag{6.63}$$

The same matrix can be assembled for all remaining elements. Note that the only metric in the matrix is the element length ζ_e. No assumption was made so far that this should be equal for all elements. Thus this formulation is general for arbitrary distribution of grid nodes. Once we have chosen the grid, we can substitute the length of each element in Equation 6.63 and obtain the specific matrix for the particular element. Those matrices can then be assembled in the global system matrix as we will see next.

But first, let’s take the specific case of Figure 6.2 where the domain was divided into equal elements of length 2 and using $\psi^2 = 3$ as in Section 3.2, we obtain the following matrix for each element:

$$\begin{bmatrix}5.2 & -4.9 \\ -4.9 & 5.2\end{bmatrix}\begin{Bmatrix}\theta_1 \\ \theta_2\end{Bmatrix} = \begin{Bmatrix}-\dfrac{d\theta}{d\zeta} \\ \dfrac{d\theta}{d\zeta}\end{Bmatrix} \tag{6.64}$$

The contributions from all elements can be assembled by placing the matrix for each element in the global matrix. This means that for internal nodes where the node is shared between two elements, the contributions need to be overlapped, or in other words added together. The process is explained schematically in Figure 6.14 for a system containing three elements with their contributions are labelled a, b and C. Each colour represents a column of the assembled matrix. Notice that the two middle columns and rows each containing contributions from two neighbouring elements.

$$\begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22}+b_{11} & b_{12} & \\ & b_{21} & b_{22}+c_{11} & c_{12} \\ & & c_{21} & c_{22} \end{bmatrix}$$

Figure 6-4: Schematic of system matrix assembly

Following the same procedure, the system matrix can be assembled for the six nodes from the contribution matrices of Equation 6.64 to give the following system matrix:

$$\begin{bmatrix} 5.2 & -4.9 & 0.0 & 0.0 & 0.0 & 0.0 \\ -4.9 & 10.4 & -4.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & -4.9 & 10.4 & -4.9 & 0.0 & 0.0 \\ 0.0 & 0.0 & -4.9 & 10.4 & -4.9 & 0.0 \\ 0.0 & 0.0 & 0.0 & -4.9 & 10.4 & -4.9 \\ 0.0 & 0.0 & 0.0 & 0.0 & -4.9 & 5.2 \end{bmatrix} \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{Bmatrix} = \begin{Bmatrix} -\dfrac{d\theta}{d\zeta} \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ \dfrac{d\theta}{d\zeta} \end{Bmatrix} \qquad (6.65)$$

Multiplying across and using the boundary conditions $\theta_1 = 1$ and $\theta_6 = 0$ we obtain the following system of linear equations:

$$\begin{aligned} &\theta_1 = 1 \\ &-4.9\theta_1 + 10.4\theta_2 - 4.9\theta_3 = 0 \\ &-4.9\theta_2 + 10.4\theta_3 - 4.9\theta_4 = 0 \\ &-4.9\theta_3 + 10.4\theta_4 - 4.9\theta_5 = 0 \\ &-4.9\theta_4 + 10.4\theta_5 - 4.9\theta_6 = 0 \\ &\theta_6 = 0 \end{aligned} \qquad (6.66)$$

Comparing the system of Equations in Equation 3.23 and Equation 6.66, we notice that they are essentially the same set of equations. This can be assured by multiplying all terms on the left and right of Equation 6.66 by (-5.1). We obtain Equation 3.23. This means that the solution for this system is the same as Equation 3.23.

This leads to the verification of Equations 6.66 as the solution for Equations 3.23 was equivalent to the analytic solution. It also leads us to the conclusion that the discretisation using linear one-dimensional elements is equivalent to the second order accurate central difference scheme of the Finite Difference formulation if we used equal grid spacing. However, the important point is that the Finite Element scheme can be used in a straight forward manner using irregular grids. In additions, the incorporation of higher order schemes by using higher order elements is straightforward.

You might think that the derivation of the Finite Element formulation is cumbersome compared to the Finite Difference formulation, and at the end, we obtained the same results as the simple finite difference scheme. That is correct. However, the advantages of the Finite Element formulations become more apparent in two and three dimensional problems when arbitrary geometries are handled.

7 The Finite Volume Method

The Finite Volume method was introduced in Chapter 3. It was mentioned that it can be viewed as a special case of the weighted residual method, were the weighting function is 1. We also presented an alternative way of obtaining a Finite Volume discretisation using the integral form of the differential equation.

The solution domain needs to be divided into non-overlapping cells surrounded by boundary edges in two-dimensions or boundary faces in three-dimensions. Integrating by parts leads the volume integral to equal a flux through the volume boundary. Working out these fluxes in terms of the unknown field variables at grid points either at the cell corners or centres leads to a system of algebraic equations which can be solved for the unknown field variables.

In this Chapter, we will explain the principles of the Finite Difference Method through worked examples that start by one-dimensional simple models. We will build these up gradually to more complex models and multiple dimensions. We will also compare the discretisation resulting from the Finite Volume formulation to that of the Finite Difference and Finite Element formulations to the fin problem.

7.1 The diffusion equation

To illustrate the basic concepts of the Finite Volume discretisation for fluid flow problems. Let's consider the diffusion equation. This equation is known as the Stokes equation, which contains the pressure and viscous terms in the Navier-Stokes equations. In one-dimensional steady state formulation, it takes the form:

$$\frac{d}{dx}\left(\mu\frac{du}{dx}\right)-\frac{dp}{dx}=F \tag{7.1}$$

where F is the body force. To simplify matters further, we will ignore the pressure term for the moment. Once this is done, the equation resembles the heat conduction equation which also includes a source term. To be more physically meaningful in this analysis, let's consider the heat conduction equation instead which takes the form:

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right)+S=0 \tag{7.2}$$

Here, k is the thermal conductivity, S is the source or internal heat generation per unit volume and T is the temperature.

To derive a Finite Volume discretisation, we will use a three point stencil as shown in Figure 7.1. We are seeking to derive the discretisation for the middle point *P*. We have also used the conventional Finite Volume notation for the surrounding point of *E* and *W* for east and west.

The next stage is to to integrate Equation 7.2 over the cell (volume) which is highlighted in Figure 7.1, this gives:

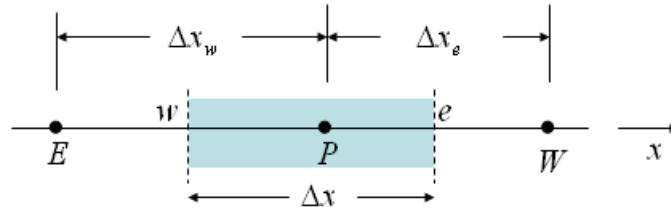



Figure 7-1 Finite Volume stencil for the 1D conduction equation

$$\int_w^e \frac{d}{dx}\left(k\frac{dT}{dx}\right)dx + \int_w^e S\ dx = 0 \tag{7.3}$$

Giving:

$$\left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + \int_w^e S\ dx = 0 \tag{7.4}$$

To enable the calculation of the temperature gradient at the east and west boundaries of the cell, we need to make an assumption of the temperature profiles within the grid. A reasonable assumption would be that the temperature is varying linearly between grid points as shown in Figure 7.2. This allows a straight forward evaluation of the first two terms of Equation 7.4.

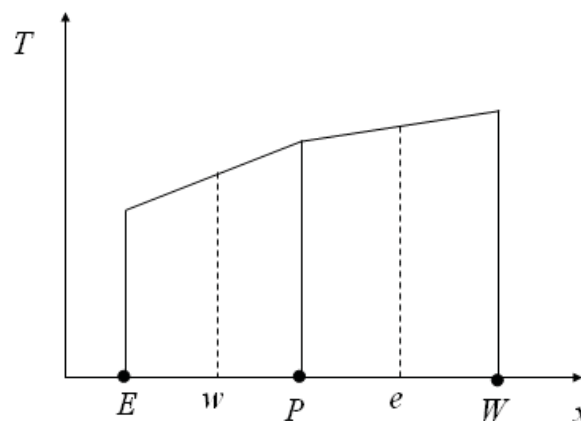



Figure 7-1: Piecewise liner profile

For the source term, we will assume that the average value $\overline{S}$ prevails over the control volume. Thus:

$$\frac{k_e(T_E - T_P)}{\Delta x_e} - \frac{k_w(T_P - T_W)}{\Delta x_w} + \overline{S}\Delta x = 0 \tag{7.5}$$

Example 7.1

Apply the discretisation used in Equation 7.5 for Equation 3.21: $\frac{\partial^2\theta}{\partial\zeta^2} - \psi^2\theta = 0$ for $\psi^2 = 3$ and equal grid spacing of 0.2.

Solution:

The discretisation applied to Equation 3.21 gives:

$$\frac{(\theta_E - \theta_P)}{\Delta x_e} - \frac{(\theta_P - \theta_W)}{\Delta x_w} - 3\theta_P\Delta x = 0$$

For equal grid spacing, $\Delta x = \Delta x_e = \Delta x_w = 0.2$, thus dividing by Δx and rearranging gives:

$$\frac{(\theta_E - 2\theta_P + \theta_W)}{(\Delta x)^2} - 3\theta_P = 0$$

$$\frac{(\theta_E - 2\theta_P + \theta_W)}{(0.2)^2} - 3\theta_P = 0$$

$$25\theta_E - 53\theta_P + 25\theta_W = 0 \qquad (7.6)$$

Equation 7.6 is the same as Equation 3.22. Thus the finite volume discretisation in Equation 7.6 is equivalent to the 2nd order Finite Difference discretisation on the same stencil. We also found in Chapter 6, for the same example that this is also equivalent to the Finite Element discretisation over the same grid size.

In general, in the Finite Volume discretisation, it is not necessary that the distances Δx_e and Δx_w be equal. In fact, the use of non-uniform grid spacing is often desirable as it enables the effective use of computing power. In general, an accurate solution will be obtained when the grid is sufficiently fine. However, there is no need to use fine grids in regions where the field variable changes slowly with the space coordinate. On the other hand, fine grids are required when the variation is steep.

Similar equations can be formulated for all internal cells as discussed above. For boundary points, the last cell is not complete as shown in Figure 7.3a. Boundary conditions need to be applied. This is done depending on the type of boundary conditions.

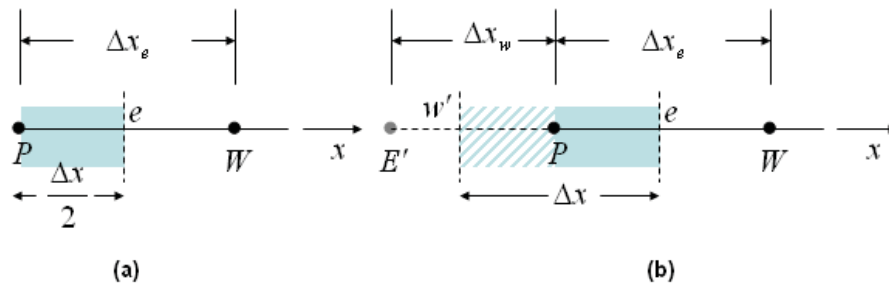



Figure 7-3: Application of boundary conditions

If the value of the field variable is given, then there is no need to formulate a flux calculation there as the equation for that cell will be replaced by the given temperature for the end node. For a give gradient of the flow field, a treatment similar to that performed with the Finite Difference methods in Chapter 5 is performed.

The approach proposed here, which ensures second order accuracy of the boundary conditions discretisation is to extend the domain by an additional virtual node which completes the cell as shown in Figure 7.3b. The value of the field variable at the virtual node is computed in terms of the two neighbouring internal nodes using the inner cell scheme. Once this is done, an equation for the boundary node can be established as in Chapter 5.

7.2 The Convection Diffusion Equation

In the previous section we have seen how to treat the heat conduction terms. These are similar to the diffusion terms in the flow equations. Flow equations however contain convection terms in addition to pressure and source terms. We will keep the simple approach to explain the basic principles here by presenting the treatment of a simplification to the flow equations by including the convection and diffusion terms only. Thus the equation we will consider is:

$$\frac{d}{dx}(\rho\, u\, \theta) = \frac{d}{dx}\left(\Gamma \frac{d\theta}{dx}\right) \tag{7.7}$$

In Equation 7.7, θ represents any flow variable such as velocity in a given direction or temperature and Γ is the diffusion coefficient of the term, which represents viscosity for the momentum equation or conductivity for the energy equation. The term u on the left hand side indicates that we are considering convection by this component of the velocity field.

Let's consider the stencil in Figure 7.1 and integrate the equation for the highlighted volume as we did for the conduction equation:

$$\int_w^e \frac{d}{dx}(\rho\, u\, \theta)\, dx = \int_w^e \frac{d}{dx}\left(\Gamma \frac{d\theta}{dx}\right) dx \tag{7.8}$$

which results:

$$(\rho\, u\, \theta)_e - (\rho\, u\, \theta)_w = \left(\Gamma \frac{d\theta}{dx}\right)_e - \left(\Gamma \frac{d\theta}{dx}\right)_w \tag{7.9}$$

We have seen in the previous section how to represent the right hand side diffusion term on the given stencil. For the advection terms on the left hand side, a straight forward treatment can be easily explained if we assume equal grid spacing by working out average values of the field variable at the cell boundaries. That is:

$$\theta_e = \frac{1}{2}(\theta_E + \theta_P) \tag{7.10}$$

$$\theta_w = \frac{1}{2}(\theta_P + \theta_W) \quad (7.11)$$

Note that the assumption of equal grid spacing is done for simplicity of presentation and any grid spacing can be used provided we used the correct interpolation weighting. Substituting from Equations 7.10 and 7.11 into the left hand side of Equation 7.9, the advection term takes the form:

$$\frac{1}{2}(\rho\, u)_e(\theta_E + \theta_P) - \frac{1}{2}(\rho\, u)_w(\theta_P + \theta_W)$$

To cast more light on this discretisation, let's assume a scenario where ρu is constant over the stencil; that is, $(\rho u)_e = (\rho u)_w = (\rho u)$ the above formulation for the advection term becomes**:**

$$\frac{1}{2}(\rho\, u)(\theta_E + \theta_P) - \frac{1}{2}(\rho\, u)(\theta_P + \theta_W) = \frac{1}{2}(\rho\, u)(\theta_E - \theta_W)$$

This shows that the scheme reduces to a second order central difference scheme for the advection term. It was demonstrated in Section 4.1 that the use of this scheme for advection terms leads to an unstable solution which is not a desirable outcome.

For this reason, other schemes need to be considered for the advection terms. The upwind scheme is a possible alternative which will be discussed in the following subsection.

7.2.1 The Upwind Scheme

It was shown in Section 4.1 that a one sided difference scheme for the advection term leads to a conditionally stable solution. However, it was noticed in Figure 4.2 that this scheme leads to a solution which captures the shape of the convected wave, but not its amplitude. The convected wave seems to have undergone a diffusion process in addition to the convection. We will explain the reason behind this here and examine a way of getting a higher order stable solution for the advection term while avoiding this spurious diffusion.

To do this, the upwind scheme, which is a one sided scheme will be introduced first. In the upwind scheme, the value of the field variable at the interface is calculated from the upwind side of the cell, that is:

$$\theta_e = \theta_P \quad \text{if} \quad u > 0 \quad (7.12)$$

$$\theta_e = \theta_E \quad \text{if} \quad u < 0 \quad (7.13)$$

$$\theta_w = \theta_W \qquad \text{if} \qquad u > 0 \tag{7.14}$$

$$\theta_w = \theta_P \qquad \text{if} \qquad u < 0 \tag{7.15}$$

Thus for a positive value of u, the convective term of Equation 7.9 can be written as:

$$(\rho\, u)_e \theta_P - (\rho\, u)_w \theta_W \tag{7.16}$$

If we again imagine a case with constant value of $(\rho\, u)$ over the stencil, we notice that this scheme is equivalent to a first order left hand side scheme which produced the conditionally stable solution in Section 4.1, despite, as mentioned above, the spurious dissipation. To understand the source of the spurious dissipation, lets assume, without loss of generality, that $(\rho\, u) = 2$ to simplify the analysis. The upwind scheme in 7.16 can be written as follows:

$$2\theta_P - 2\theta_W = \theta_P + \theta_P - \theta_W - \theta_W$$

Let's add and subtract θ_E from the right hand side:

$$\theta_P + \theta_P - \theta_W - \theta_W = \theta_P + \theta_P - \theta_W - \theta_W + \theta_E - \theta_E$$

And now regroup the terms on the right hand side to give:

$$(\theta_E - \theta_W) - (\theta_E + \theta_W - 2\theta_P) \quad (7.17)$$

The first term in the brackets in Equation 7.17 is proportional to a central difference second order scheme of the first derivative of the field function while the second term is proportional to the second derivative of the field function. Hence the first order upwind scheme for the advection term seems to be a discretisation of the term:

$$\frac{d\theta}{dx} + \alpha \frac{d^2\theta}{dx^2} \quad (7.18)$$

An unstable central difference scheme has thus been stabilised by the addition of a diffusion term. The diffusion term is not physical, but rather numerical. The second conclusion is that the reason behind the dissipation of the solution when a first order one-sided scheme was used is the implicitly added numerical dissipation.

This concept led researchers to develop various other schemes by using the unstable central difference scheme with the addition of numerical diffusion, which is also known in the literature as the artificial dissipation. The basic concept behind these schemes is to add a small amount of dissipation sufficient to stabilise the scheme while trying to minimize the non-physical spreading, or dissipation of the solution.

The early schemes of this type are called the Lax-Windroff schemes. The interested reader can refer to Hirsch (1990) for more details.

7.3 Extension to multi-dimensional problems

The extension to two and three dimensional problems follows the same principles. The same logic of using the simple conduction problem will be used to illustrate the principles. So let's consider the two dimensional steady state conduction problem:

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right) + \frac{d}{dy}\left(k\frac{dT}{dy}\right) + S = 0 \quad (7.19)$$

Assume we require solving this equation over a domain Ω with boundary Γ as shown in Figure 7.3. To do this using a Finite volume method, we need to integrate over the domain, hence:

$$\int_{\Omega}\left[\frac{d}{dx}\left(k\frac{dT}{dx}\right)+\frac{d}{dy}\left(k\frac{dT}{dy}\right)\right]d\Omega+\int_{\Omega}Sd\Omega=0 \qquad (7.20)$$

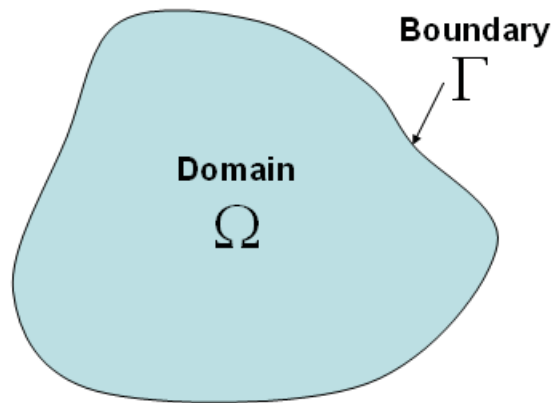



Figure 7-3: Solution domain and boundary

The left hand integral in Equation 7.20 is equivalent to a surface integral as follows:

$$\int_{\Omega}\left[\frac{d}{dx}\left(k\frac{dT}{dx}\right)+\frac{d}{dy}\left(k\frac{dT}{dy}\right)\right]d\Omega=\oint_{\Gamma}\left[k\frac{dT}{dx}+k\frac{dT}{dy}\right]d\Gamma \qquad (7.21)$$

Thus the volume integral has been replaced by an integral of the fluxes through the boundary of the domain. If the solution domain is divided to a large number of non-overlapping volumes, then Equation 7.21 can be applied to each volume in turn which will result in the required discretisation.

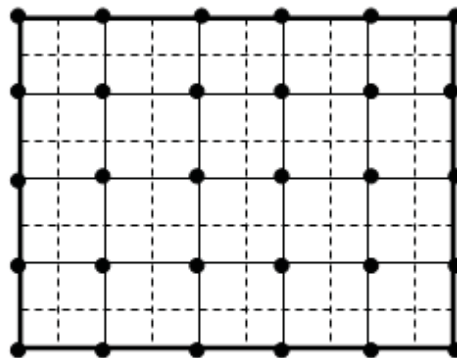



Figure 7-4: Finite volume grid and cells.

Let's take for example a simple rectangular domain as shown in Figure 7.4. This domain is discretised using a regular grid as shown in the figure. The grid points are at the vertices of the mesh. The finite volume cells can be laid over the grid as shown using the dotted lines. A stencil containing on grid cell is shown enlarged in Figure 7.5.

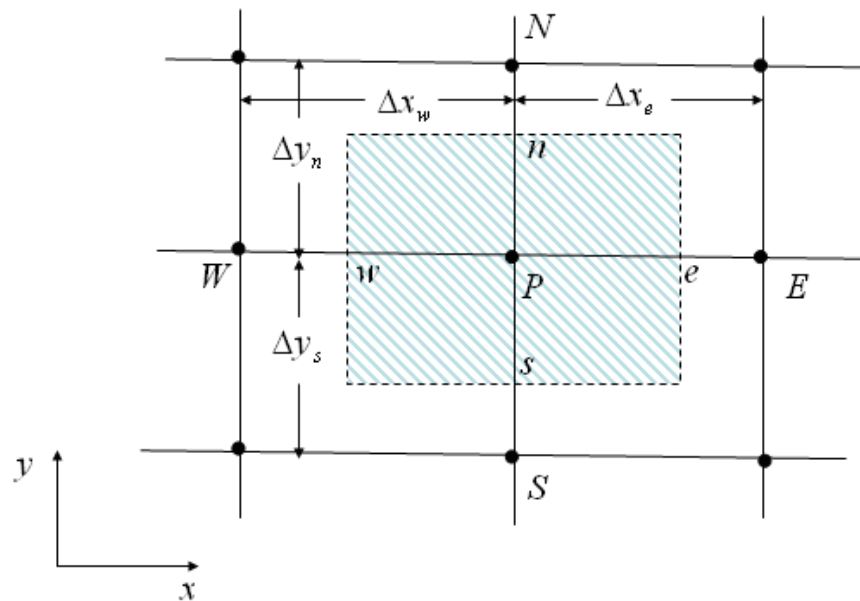



Figure 7-5: Two dimensional Finite Volume stencil

A notation similar to that used in the one-dimensional case is used here for clarity. Note the addition of two extra boundaries to the Finite Volume cell designated N and S for North and south, which also required the inclusion of two additional points in the stencil. No assumptions were made on the positions of the cell boundaries, so they can be any where between the surrounding nodes and the node of concern.

The boundary edges of the cell, however, were deliberately aligned with the coordinate axis to simplify the integration. This is not a restriction on the Finite Volume method as the boundary surfaces can take any orientation with regards to coordinate axis as long as the flux through the boundary is calculated correctly. However, this simplification was used here to illustrate the principles of the Finite Volume method.

The boundary integral on the right hand side of Equation 7.21 can now be broken into four parts, each will be integrated independently. The summation of all the four integrals will produce the total required flux. So these four fluxes are:

East side: $\frac{k_e(T_E - T_P)}{\Delta x_e}$

West side: $\frac{k_w(T_P - T_W)}{\Delta x_w}$

North side: $\frac{k_n(T_N - T_P)}{\Delta y_n}$

South side: $\frac{k_s(T_P - T_S)}{\Delta y_s}$

If we use the average source term $\bar{S}$ as we did in the one-dimensional case, then the discrete form of Equation 7.20 for the grid stencil in Figure 7.5 will take the form:

$$\frac{k_e(T_E - T_P)}{\Delta x_e} + \frac{k_w(T_P - T_W)}{\Delta x_w} + \frac{k_n(T_N - T_P)}{\Delta y_n} + \frac{k_s(T_P - T_S)}{\Delta y_s} + \bar{S}\Delta x \Delta y = 0 \qquad (7.22)$$

Exercise:

For a regular grid of $\Delta x = \Delta y$ and all cell boundaries located at mid-distance between points, show that the scheme in Equation 7.22 is equivalent to a second order accurate Finite Difference discretisation in two dimensions.

The above discretisation can be applied to all internal cells obtaining a similar equation for each node. Boundary node treatment depends again on the type of boundary conditions. For given temperatures at the boundaries, a straight forward addition of the equations for these nodes is performed. For given heat flux (Temperature gradient), a virtual point can be added opposite each point on the boundary and a treatment similar to that described in the one dimensional case can be performed.

Once the system of equations is obtained, it can be solved for the unknown nodal temperatures. For the more complicated flow equations containing advection and diffusion terms, treatment of the advection terms is a straightforward extension to that used in the one dimensional case.

7.4 Unstructured grids

With irregular grids such as the one shown in Figure 7.6 a similar procedure to that described above for regular grids can be used. Two approaches are possible. The first is the cell-centred approach. In this case, the flow variables are sought at nodes at the centre of the cell and the fluxes are computed through the edges or faces of the cell. In the second approach, a vertex centred approach is used. This is illustrated in Figure 7.6 where cells are constructed around the grid nodes by joining the centres of each neighbouring element around the node. These cells are shown using the dotted lines. The fluxes are then computed through those new edges or faces of the cell.

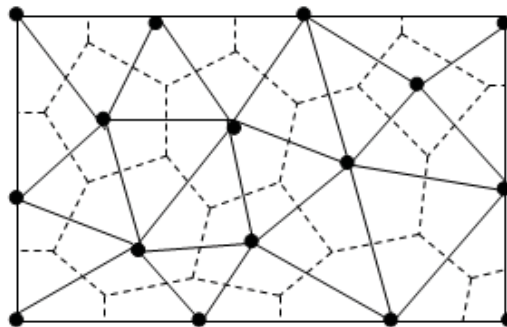



Figure 7-6: Finite volume cells on an unstructured grid

It is important to note here that in the two-dimensional case, each face contributes to the flux in both coordinate directions as the normals to the cell faces, in the general case, are not aligned with the coordinate directions.

In addition, care in the numerical scheme should be exercised in calculating the cell volume, which is required for the source terms, for example. Because of the irregular shape of the grid, one way of ensuring the correct calculation of the cell area (volume) in two-dimensions is to divide it to triangles, calculate the area of each triangle and then adding them up.

In three dimensions, the process can be done similarly by dividing the cell to tetrahedral cells and calculating the volume of each one separately, then adding them up.

8 Solution Methods Systems of Equations

In the previous three Chapters, we discussed the basic discretisation methods for the fluid flow equations. We noted various similarities and differences between the three methods presented. One of the common features of all the discretisation methods is that they lead to a system of algebraic equations that needs to be solved for the unknowns at the grid points.

The system of equations can be linear or nonlinear depending on the governing equations and the discretisation procedure. We may choose to resort to standard solution procedures such as the direct inversion of the system matrix. However, fluid flow problems can contain very large number of grid points making direct equation solvers an impractical approach.

To illustrate the problems encountered, consider a typical system of equations resulting from the discretisation of steady flow equations. The system can be expressed in matrix notation as:

$$[A]\{x\} = \{B\} \tag{8.1}$$

where the matrix A is the system matrix. For nonlinear systems, this matrix will be a function of the field variable x. The vector B is the right hand vector which typically contains the boundary conditions and in some situations the source terms. For structured grid problems encountered when the Finite Difference Method or Finite volume method is used, the system of equations is banded with the rest of the terms being zero. An example is shown by Equation 8.2, where the x's show non-zero terms.

$$\begin{bmatrix} x & x & & & & & & \\ x & x & x & & & & & \\ & x & x & x & & & & \\ & & x & x & x & & & \\ & & & x & x & x & & \\ & & & & x & x & x & \\ & & & & & x & x & x \\ & & & & & & x & x \end{bmatrix} \tag{8.2}$$

In this case, it will not be a wise choice to store the entire matrix or choose an approach by which the solution is obtained by direct inversion of the matrix, i.e. as shown by Equation 8.3.

$$\{x\} = [A]^{-1}\{B\} \tag{8.3}$$

In particular, for large problems which might contain several millions of grid points, this approach becomes impractical in terms of both storage and computing time. The storage problem might be eliminated by recognising that the matrix is banded and then choosing an appropriate storage strategy, but the inversion computing requirements will still be high.

The storage requirements become even higher if an unstructured grid is employed using either Finite Element of Finite Volume methods. In this case, because of the random point distribution, the system matrix will typically be sparse. An example is shown in Equation 8.4.

$$\begin{bmatrix} x & & x & & & x & & \\ x & x & & & x & & & \\ & x & & x & & & x & \\ & & x & x & x & & & \\ x & & & x & & x & & \\ & & x & & & x & x & \\ & x & & x & & x & x & x \\ x & & & & & & x & x \end{bmatrix} \tag{8.4}$$

Again here some storage scheme might be devised which avoids the storage of zero entries. For example, nonzero entries and their address can be stored. But again the inversion process will still be expensive.

For this reason, alternative approaches to the direct solution method will be required. In this Chapter, we will introduce some of these methods.

8.1 Nonlinear Systems

The convective terms in the flow equations lead to nonlinear systems where the matrix is a function of the flow variables such as that shown in Equation 8.5.

$$[A(x)]\{x\} = \{B\} \tag{8.5}$$

The nonlinearity can also arise from the dependence of the fluid properties on the flow variables, for example, if the viscosity is a function of temperature. There are several methods that can be used to solve such problems. We will discuss two of them here.

8.1.1 Newton's Method

Newton's method can be used for the solution of nonlinear systems as well as linear systems. The starting point is to assume an initial guess of the solution $\{x\}$ and then express Equation 8.5 using this solution as follows:

$$\{R\} = [A(x)]\{x\} - \{B\} \tag{8.6}$$

where R is a residual which arises from the fact that the initial guess does not satisfy the equation. Note that by performing the multiplication $[A(x)]\{x\}$, a vector rather than a matrix results. So if during the discretisation this vector is assembled term by term, there is no need to store a system matrix and this reduces the storage requirements dramatically.

The Newton's method is an iterative process which is based on the expression in Equation 8.6 which can be written as follows:

$$\{x^{n+1}\} = \{x^n\} - [J^n]^{-1}\{R^n\} \tag{8.7}$$

where the superscript n indicates current iteration level and $n+1$ indicates the new iteration level. The Jacobian J is obtained from:

$$J = \frac{\partial R}{\partial x} \tag{8.8}$$

Note that that the resulting system in Equation 8.7 is a system of linear equations which still needs to be solved. The advantage of Newton's method is that it accelerates the convergence of nonlinear systems, particularly, if the initial solution is close to the actual solution.

The main computational cost in this method is the requirement to inverse the Jacobian matrix. However, iterative approaches will be explained in Section 8.2 which will eliminate this requirement. A schematic diagram of the Newton's method is shown in Figure 8.1

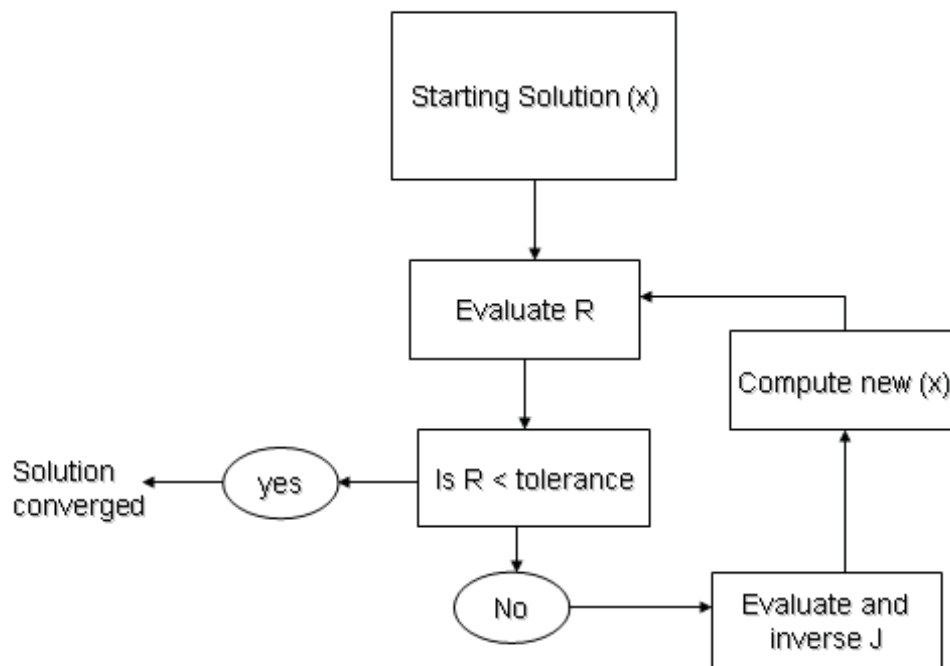



Figure 8-1: Schematic diagram for Newton's method

8.1.2 Quasi-Newton Method

As mentioned above, the most expensive part of Newton's method is the assembly and inversion of the Jacobian matrix. An alternative which relaxes this restriction is the use of the Quasi-Newton method. In this method, Equation 8.7 is replaced by:

$$\{x^{n+1}\} = \{x^n\} - \omega^n [H^n]\{R^n\} \tag{8.9}$$

where ω is some scalar. The matrix $[H]$ is an approximation of the inverse of the Jacobian matrix. This matrix is systematically modified during the iteration process until it approaches the inverse of the Jacobian matrix on convergence with the appropriate choice of the scalar ω. The matrix $[H]$ is obtained iteratively using the following procedure. First an initial value of $[H]$ is assumed at time n and then an iterative process is set as follows:

$$\{P^n\} = -\omega^n [H^n]\{R^n\}$$

$$\{\alpha^n\} = \frac{1}{\{P^n\}^T \left(\{R^{n+1}\} - \{R^n\}\right)}$$

$$[H^{n+1}] = [H^n] - \{\alpha^n\}\left([H^n]\left(\{R^{n+1}\} - \{R^n\}\right)\{P^n\}^T + \{P^n\}\left(\{R^{n+1}\} - \{R^n\}\right)[H^n]\right)$$

Although the Jacobian matrix is sparse, the matrix $[H]$ is dense. In a typical iteration procedure, the initial guess for the matrix $[H]$ is taken as the identity matrix.

8.2 Methods for the solution of linear systems

Linear systems can arise from the discretisation of differential equations that do not have nonlinearities, such as the heat conduction equations or similar diffusion problems. They also can result from linearisation of nonlinear systems as described in the previous section.

The system of algebraic equations takes the form shown in Equation 8.1 with the coefficient matrix $[A]$ being independent of the field variable. Gauss elimination can be used for the solution of this system. However, the efficient implementation of this method depends on the properties of the matrix $[A]$.

The Algorithm used in the Gauss elimination will depend on the nature of the matrix, for example, for a dense matrix with few nonzero coefficients, the matrix is decomposed by what is known as LU decomposition. This means that the matrix is factorised to an upper and lower triangular matrices such that:

$$[L][U] = [A] \tag{8.10}$$

Where L and U matrices take the form:

$$[L] = \begin{bmatrix} x & & & & & & & \\ x & x & & & & & & \\ x & x & x & & & & & \\ x & x & x & x & & & & \\ x & x & x & x & x & & & \\ x & x & x & x & x & x & & \\ x & x & x & x & x & x & x & \\ x & x & x & x & x & x & x & x \end{bmatrix} \quad , \quad [U] = \begin{bmatrix} x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & & x & x & x & x \\ & & & & & x & x & x \\ & & & & & & x & x \\ & & & & & & & x \end{bmatrix}$$

Then the factorised form of the matrix is solved as:

$$[U]\{x\} = [L]^{-1}\{B\} \tag{8.11}$$

Because the matrix $[U]$ is an upper triangular, only back substitution is required once the right hand side is found. So the main computational cost lies in the factorisation and inverting the matrix $[L]$.

If the matrix is banded such as the tridiagonal system shown in Equation 8.2, then usually what is know as the Thomas algorithm is used. Tridiagonal matrices result from regular grid finite difference or Finite volume discretisation. The Thomas algorithm is a variant of the Gauss elimination.

The Thomas algorithm is performed using two steps. To illustrate this method, consider the tridiagonal system shown in Equation 8.12:

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & a_4 & b_4 & c_4 \\ & & & a_n & b_n \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_n \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_n \end{Bmatrix} \tag{8.12}$$

The first step is to reduce the system using the relations:

$$c_1' = \frac{c_1}{d_1}, \qquad d_1' = \frac{d_1}{d_1} \qquad \text{for the first row}$$

$$c_i' = \frac{c_i}{b_i - a_i c_{i-1}'}, \qquad d_i' = \frac{d_i - a_i d_{i-1}'}{b_i - a_i c_{i-1}'} \quad \text{for the rest of the rows}$$

The system then reduces to:

$$\begin{bmatrix} 1 & c_1' & & & \\ & 1 & c_2' & & \\ & & 1 & c_3' & \\ & & & 1 & c_4' \\ & & & & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_n \end{Bmatrix} = \begin{Bmatrix} d_1' \\ d_2' \\ d_3' \\ d_4' \\ d_n' \end{Bmatrix}$$

Then the substitution stage takes the form:

$$x_n = d_n'$$

$$x_i = d_i' - x_{i+1} c_i'$$

Note that the last equation starts from the end of the matrix and goes upwards.

The Thomas algorithm can be generalised to banded matrices of larger bandwidth resulting from discretisation of two and three dimensional problems.

8.3 Iterative methods

It was mentioned in Section 8.1 that direct solution methods can be computationally prohibitive for many CFD problems because of the size of the matrices involved. Thus it is more customary now to resort to iterative solvers. These provide two advantages. The first is that the memory requirements are significantly reduced because the system matrix is not stored. The second is the reduced computational effort.

Although iterative methods are generally thought of as being designed to nonlinear systems, they are also used for liner systems or nonlinear systems that are linearised using procedures such as Newton's Method.

Iterative methods can be view as a process of starting with an initial solution, or initial guess. This is some times obtained by interpolating boundary conditions to all internal nodes. This solution is then successively modified using some iterative approach until the solution is converged. Convergence is achieved when the difference of the solution between iteration and the previous number of iterations reaches a pre-defined tolerance, which does not increase with more iteration.

8.3.1 Basic iterative methods

These are the first methods that were developed for iterative solvers. They are closely related to simple time marching and work as the approach to steady-state solutions using finite time steps. The first method to take this form is the Jacobi method. Consider the matrix Equation 8.1 $[A]\{x\}=\{B\}$. The Jacobi method starts by assuming that an initial estimated solution x_i^n is known, where the superscript n indicates the time level. As mentioned earlier, this initial solution can be interpolated from boundary conditions, or used from the values at any boundary of the domain.

An iterative procedure is then set up as follows

$$x_i^{n+1} = \frac{B_i - \left(\sum_{\substack{j \\ j \neq i}}^{n} A_{ij} x_j^n \right)}{A_{ii}} \qquad (8.11)$$

The first important feature of the Jacobi method is that if the right hand side vector $A_{ij} x_j^n$ is assembled directly during the calculations, there is no need to store the entire matrix. The second feature is that, two solution vectors need to be stored because the solution at the current iteration level depends on the solution in the previous iteration.

Although he Jacobi method is simple, it is generally expensive computationally because it requires too many iterations for the solution to reach convergence.

A direct improvement to the Jacobi method is provided by the Gauss-Seidel method. This is similar to the Jacobi method, with the exception that the values of the solution vector x_i^{n+1} are used on the right hand side of Equation 8.11 as they become available. Thus the solution becomes:

$$x_i^{n+1} = \frac{B_i - \left(\sum_{j=1}^{i-1} A_{ij} x_j^{n+1} \right) - \left(\sum_{j=i+1}^{n} A_{ij} x_j^n \right)}{A_{ii}} \qquad (8.12)$$

Gauss-Seidel method can be as much as twice faster than the Jacobi method. Both Jacobi and Gauss-Seidel method will definitely converge if the matrix A is diagonally dominant.

Gauss-Seidel method can be improved using the method of successive over-relaxation. In this method, the solution vector x_i^{n+1} is evaluated as a weighted average between its value at the current time level and that at the new time level using a relaxation parameter which can be varied from problem to problem based on experience.

8.3.2 Conjugate gradient methods

Conjugate gradient methods are now widely used because of their efficiency. The conjugate gradient method was invented in 1952, but was not extensively used until the late 1970's, when it was realised that its efficiency can be significantly improved with the use of a preconditioner.

The conjugate gradient method can be viewed as a method by which an estimate of the eigenvalues of the system matrix A is found. Indeed, the iterative eigen analysis methods form the basis of many conjugate gradient schemes. There are several variants of this method. Here we will use one of these to illustrate how it works in the following procedure:

Step 1: Staring with the matrix A and the right hand side b, and an initial guess of the solution x^0 at iteration level $n = 0$.

Step 2: Compute an initial preconditioned vector $z_0 = M^{-1} r^0$ where M is an easy to invert approximation of the matrix A, the closer M to A the more effective the preconditioning is and the harder it is to invert

Step 3: Compute the initial residual $r_0 = b - Ax_0$, set $p_0 = r_0$

Step 4: Calculate the vector: $q_k = Ap_k$ for iteration k

Step 5: calculate the parameter $\alpha_k = (r_k, r_k)/(q_k, p_k)$ where for two vectors, m and n, the inner product is defined as: $(m,n) = \sum m_i n_i$.

Step 6: Calculate the variable at the new iteration level: $x_{k+1} = x_k + \alpha_k p_k$

Step 7: Calculate residual at new iteration level: $r_{k+1} = r_k - \alpha_k q_k$

Step 8: Calculate the preconditioner vector at new iteration level: $z_{k+1} = M^{-1} r_{k+1}$

Step 9: Calculate new norm: $\beta_k = (r_{k+1}, z_{k+1} / r_k, z_k)$

Step 10: Compute new parameter $p_{k+1} = z_{k+1} + \beta_k p_k$

Step 11: Advance iteration level $k = k + 1$

Step 12: Test for convergence, if achieved, exist iteration levels, otherwise, go to step 4 and repeat the process.

The discussion of the theoretical background for this procedure is involved and beyond the scope of this book. The steps above are presented only to illustrate the method. For more details, see for example Drikakis and Rider (2005).

8.4 Multigrid acceleration

Multigrid methods can be applied to both linear and non-linear systems. They work with a sequence of grids, such that the solution is sought on the finest grid. The remaining grids are coarser grids of the same solution domain ordered in a sequence where each grid is approximately half the number of grid points from the previous grid, ore twice coarser.

The concept is to use the coarse grids to damp the high frequency oscillations of the solution and extrapolate corrections to the finer grid solutions. This usually results significant acceleration of the iteration process to convergence.

To illustrate the method lets consider a number of grids arranged from grid 1 to *M*, where grid *M* is the finest. Then the solution sought is that of the system:

$$A^M x^M = b^M \tag{8.13}$$

Then an approximation to the solution x^M is provided by the solution on the next coarser grid x^{M-1}. Similarly, for any intermediate grid, the solution is sought on the next coarser grids. Or in other words, the solution on any grid x^M is considered a good approximation of the solution on the finer grid x^{M+1}.

Now, if the approximation to the solution is substituted into the equation, it will produce a residual:

$$b^{M+1} - A^{M+1} x^{M+1,c} = R^{M+1} \tag{8.14}$$

Where $x^{M+1,c}$ is the approximation to the solution on grid level M+1 obtained by interpolation from grid level m. This residual should satisfy the equation:

$$A^{M+1} W^{M+1} = R^{M+1} \tag{8.15}$$

Where W is the correction to the solution. If both the residual and the correction are smooth functions, the highest frequency that can be represented on the coarse grid is proportional to the grid size and hence, higher frequencies are damped out on coarser grids.

A typical multigrid procedure then is made of an iterative process where solutions are obtained on coarser grids, and then corrections are transformed to finer grids successively. Once the iteration level reaches the finest grid, solutions are prolongated back successively to the coarser grids and so on until the solution is converged.

9 Mesh Generation

In Chapter 2, we have seen how the flow is modelled using a system of partial differential equations. We then went on in Chapter 3 to show how this system is discretised in space to convert the differential equations to a system of algebraic equations that can be solved using computers to produce an approximate solution.

The discretisation process converts the continuous system to a discrete one, where a grid or mesh needs to be constructed such that the approximate solution is sought at the discrete grid points.

Generally speaking, there are two mesh types, characterised by the connectivity of the points. Structured meshes have a regular connectivity, which means that each point has the same number of neighbours. Unstructured meshes on the other hand have irregular connectivity where each point can have a different number of neighbours.

Figure 9.1 gives and example of each type of grid. In some cases, part of the grid is structured and part is unstructured. For example, in viscous flows, boundary layer could be structured and the rest of the domain unstructured.

(a)Structured grid b) unstructured grid

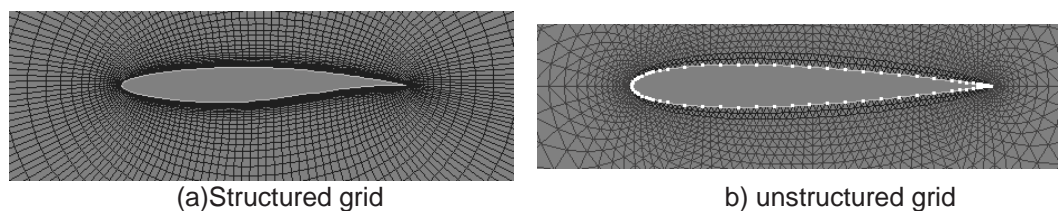



Figure 9.1: Example of structured and unstructured grid

It should be noted that a numerical algorithm need to be devised to suite the type of grid used. In most cases, numerical algorithms written to use structured grids cannot be used on unstructured grids while numerical algorithms written to use unstructured grids can use also structured grids.

In the rest of this Chapter Section 9.2 will focus on structured mesh generation methods while Section 9.3 will focus on unstructured mesh methods.

9.1 Structured girds

This section begins with a discussion of boundary fitted grids and the discretisation of partial differential equations on them, and then deals with the main methods for grid generation of simple domains. It then explains the multi-block concept for more complicated domains.

9.1.1 Boundary-fitted meshes

Structured grids are characterised by regular connectivity. This means that the points of the grid can be indexed and the neighbours of each point can be calculated rather than looked up from pre-stored data.

Grids on a rectangular domain are trivial to generate, although care should b taken near convex boundaries. More elaborate structured mesh techniques need to be used when meshing domains of irregular boundaries

Generally, the meshes are generated so that they fit the boundaries, with one coordinate surface forming one of the boundaries. This enables generating good quality mesh near the boundary and enables the use of fast solution algorithms.

The most common method of generating boundary-fitted grids is to have a continuous grid that fits all the boundaries. The effect is to fit a contiguous set of rectangular computational domains to a physical domain with curved boundaries as shown in Figure 9.2

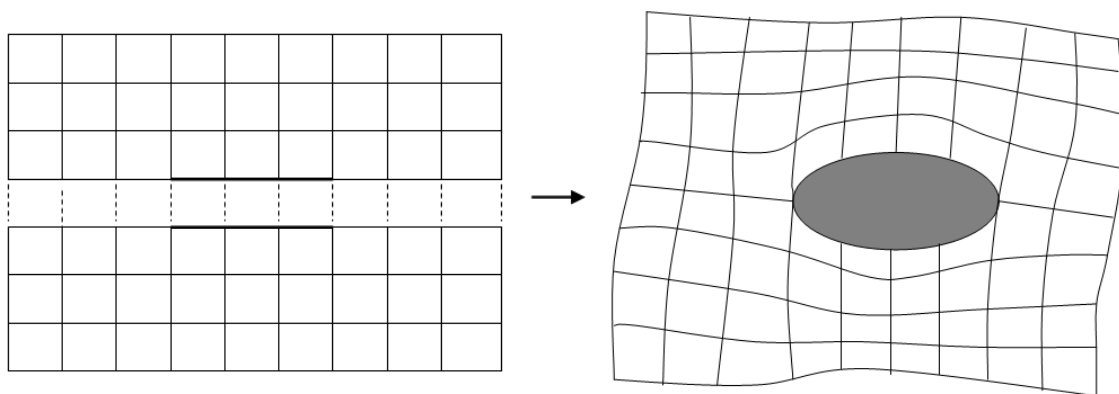



Figure 9.2: Example of boundary fitted grid

It is difficult to fit complex domains with one mapping from a rectangular grid, without generating excessively skewed grids. To get round this problem, the domain is split up into blocks and each block is gridded with some continuity requirements at the block interfaces. The decomposition of the domain into blocks is usually done manually using CAD methods.

Once the grid is generated, the physical problem has to be discretised and solved on that grid. It is possible to calculate Finite Difference or Finite Volume formulas using the physical grid directly, but this will reduce the order of accuracy of the numerical scheme.

The preferred method is to transform the equations used to model the problem into a computational space where the grid is rectangular. The resulting equations will contain the effects of the domain transformation. The finite differencing can be done on the rectangular grid. The equations are then transformed back onto the physical space. See Figure 9.3 for illustration of this process.

Although rectangular grids allow high order accuracy of the discretisation, curvilinear grids can cause some inaccuracies, such as increased numerical diffusion. Additionally, for multi-block grids, corner points are usually points with non-standard connectivity leading to complexity of the numerical scheme.

For simple domains, a single grid can be used without leading to excessive skewness. The computational space is rectangular in 2D and cuboid in 3D. A one-to-one mapping is defined from the computational space to the physical space. Several methods are currently used for this process such as:

Algebraic grid generation where the grid is interpolated from the physical boundaries.

Methods where a Partial Differential Equation (PDE) for the grid point coordinates is solved

- Variational grid generation, where some function is maximised or minimised, such as smoothness.

Conformal mapping.

Figure 9.3: Transformations between physical and computational domains

9.1.2 Algebraic grid generation

This method is also called transfinite interpolation. This method will be described here for 2D. The extension to 3D is straightforward.

Let's consider the example shown in Figure 9.3. We take the computational domain as a unit square varying from 0 to 1 in both the s and t directions. Let's also assume that the computational domain is defined using x-y coordinates. To generate a grid in the physical space, we generate a regular grid in the unit square and map this onto the physical space.

There are two basic requirements for this mapping. The first is that it must be one-to-one. The second is that the boundaries of the computational space must map into the physical space, otherwise, the mapping can be arbitrary.

In transfinite interpolation, the physical coordinates are interpolated from their value on the boundaries of the computational domain. The 2D interpolation is constructed as linear combination of two 1D interpolations, usually called projections.

It is possible to extend the transfinite interpolation so that it interpolates to several coordinate lines, not just the boundaries. This is useful in controlling the grid density and also ensures one-to-one mapping. An example of a grid generated using the transfinite interpolation is shown in Figure 9.4.

There are some problems associated with transfinite transformation. For example, the mapping will propagate singularities (corners) into the interior of the domain. Other problems might arise if the mapping is not one-to-one such as overspill as shown in Figure 9.5

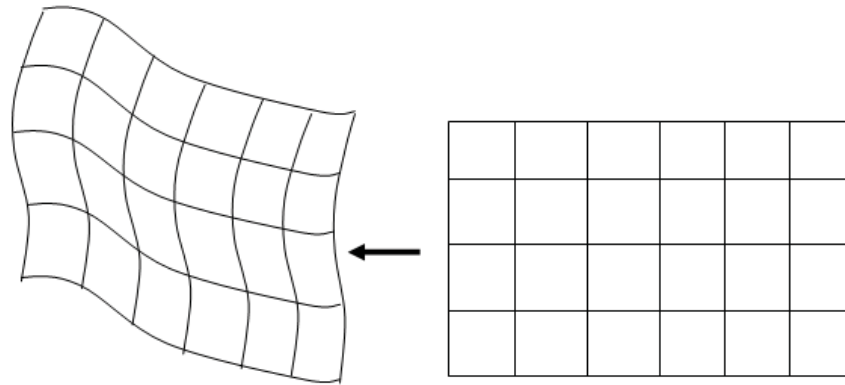



Figure 9.4 Mesh generated by transfinite interpolation

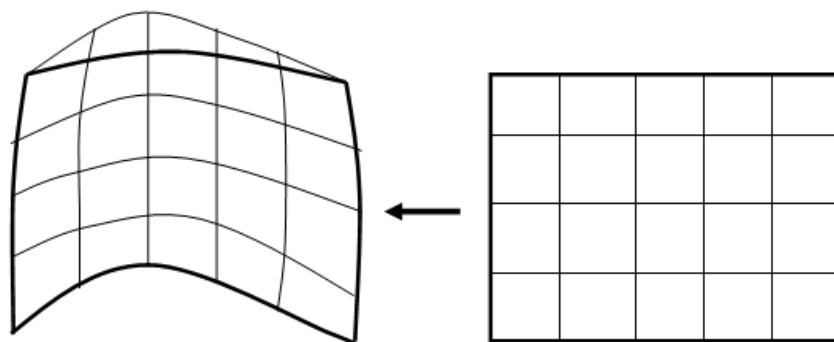



Figure 9.5 Example of overspill of mesh

9.1.3 Partial Differential equation mesh generation

The idea behind these methods is to define Partial differential equations (PDE's) for the physical space coordinates in terms of the computational space coordinates and then solve these equations on a grid in computational space to create a grid in physical space.

We are going to introduce the concept of structured mesh generation here using PDE's without going into too much detail. The interested reader can refer to specialist books for more detail.

To introduce the concept, we are going to talk first about transformation between the physical domain and the parametric domain, and the reasoning behind using this transformation.

Referring to Figure 9.3, we mentioned that if the physical domain does not align with the coordinate axis, we perform a transformation to a rectangular computational domain. This allows for generation of regular grids, which enables discretisation to retain the high order accuracy, which always reduces if distorted grids are used. Once the equations are discretised, they are transformed back to the physical space and solved.

Let's focus again on 2D (without loss of generality) and assume a domain defined in the x,y physical domain as shown in Figure 9.6. The relationship between the physical (x, y) and computational (ξ,η) will be established. The corresponding opposite transformation will also be established.

It is assumed that there is a unique, single-valued relationship between the generalised coordinates and the physical coordinates, which can be written as:

$$\xi = \xi(x, y) \quad , \qquad \eta = \eta(x, y) \tag{9.1}$$

This also implies that

$$x = x(\xi,\eta) \quad , \qquad y = y(\xi,\eta) \tag{9.2}$$

The functional relationships are determined by the mesh generation process.

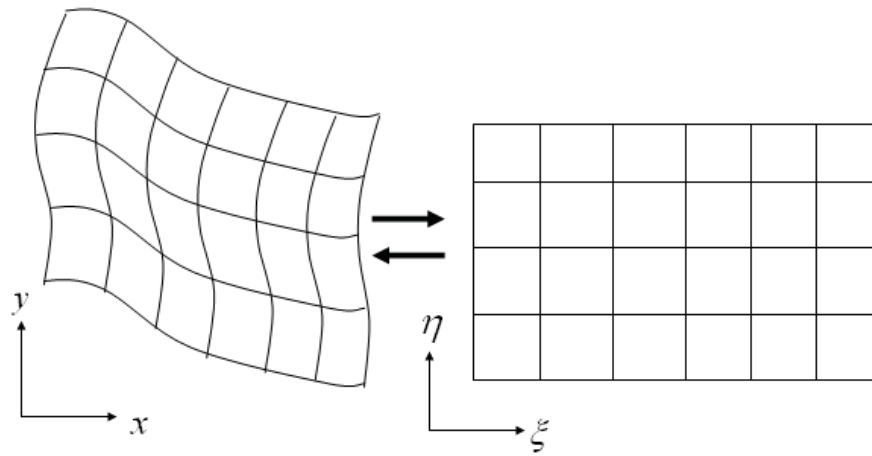



Figure 9.6: transformation between physical and parametric domains

Given these functional relationships, the governing equations can be transformed into corresponding equations containing partial derivatives with respect to the parametric space. For example the derivative of temperature with respect to x and y becomes:

$$\frac{\partial T}{\partial x} = \frac{\partial T}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial T}{\partial \eta}\frac{\partial \eta}{\partial x} \tag{9.3}$$

$$\frac{\partial T}{\partial y} = \frac{\partial T}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial T}{\partial \eta}\frac{\partial \eta}{\partial y} \tag{9.4}$$

The inverse transformation can be written as:

$$\frac{\partial T}{\partial \xi} = \frac{\partial T}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial T}{\partial y}\frac{\partial y}{\partial \xi} \tag{9.5}$$

$$\frac{\partial T}{\partial \eta} = \frac{\partial T}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial T}{\partial y}\frac{\partial y}{\partial \eta} \tag{9.6}$$

There are several techniques that can be used to generate grids based on the above. We will briefly outline a general technique based on the solution of the Poisson equations. This technique does not necessarily produce conformal or orthogonal grids, but allows more control over the clustering of interior grid points.

The technique is based on the concept that interior point grid generation can be presented as a boundary value problem, preferably in the parametric domain. A partial differential equation is then solved to specify the grid point locations. The Poisson equation solved is of the form:

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = P(\xi,\eta) \quad , \qquad \frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = Q(\xi,\eta) \tag{9.7}$$

Where P and Q are pre-defined functions used to control grid clustering.

A rectangular grid is generated in the parametric domain first. Equation 9.7 is discretised using the techniques described in Chapter 3. This results in a system of algebraic equations which can be solved numerically to obtain the grid points in the physical domain. For more details, the interested reader can refer to Fletcher (1991) for example.

It should be noted that the use of regular structured grids dominated CFD methods in the early developed CFD codes. This was dictated by the relatively efficient computational storage and solution times required by these methods compared to unstructured grid. However, the significant amount of human effort required during the process of mesh generation, and the continuous increase in computational power and cheap memory shifted the interest to unstructured grids which can be generated in a more automated fashion even for complex geometries.

9.2 Unstructured grids

Unstructured meshes have been developed mainly for Finite Element Methods. There is a large range of possible shapes for Finite Elements: tetrahedral, prisms and bricks and there can be arbitrary connectivity leading to unstructured meshes. Although it is possible to automatically generate any type of unstructured grids, it is much easier to generate tetrahedral grids.

Brick elements might be desirable, particularly in boundary layers. They are much harder to generate and there are not many algorithms that can generate this type of elements automatically for complex geometries.

Although unstructured grids were mainly developed for Finite Element Methods, there are many Finite Volume algorithms nowadays that discretise the flow equations over unstructured grids. The mesh requirements for Finite Element and Finite Volume methods are:

The mesh must be valid in the sense that it should have no holes, no self intersections, and no faces joined at two or more edges.

The mesh must conform to the boundary of the domain.

The density of the mesh must be controllable to allow a balance between solution accuracy and computational requirements.

Gird density will vary according to the local accuracy requirements. However, grid density variation must be smooth to reduce numerical diffusion errors.

Some solution algorithms have requirements on the shape of elements or grid cells that can be handled.

In 2D and for boundary faces in 3D, the most used types of elements are triangles and quadrilaterals. For 3D geometries, the most used types are Tetrahedral, Hexahedral, prisms and sometimes, polyhedral elements or cells.

There are several methods for the generation of unstructured grids. The most popular methods will be discussed in the following sub-sections.

9.2.1 Surface meshing

Most unstructured mesh generators start by generating a mesh on the surfaces bounding the geometry, then a 3D mesh is generated inside the domain. For geometries with a parametric representation, a general surfaces is made of patches, each have 4 sides. These are then divided to smaller quadrilateral patches as shown in Figure 9.6(a). These are defined by parametric lines along two orthogonal directions, *u* and *v*.

A simple way of generating a mesh on this type of surfaces is first to map the surface onto a 2D parametric surface as shown in Figure 9.6(b). Mapping relations such as those described by Equations 4.1 and 4.2 can be used to map forward and backward between the physical surface and the parametric surface.

An unstructured mesh can easily be generated in the 2D mapped surface using one of the methods described below. Once the mesh is generated, it is mapped to the physical surface as illustrated in Figure 9.7

(a) (b)

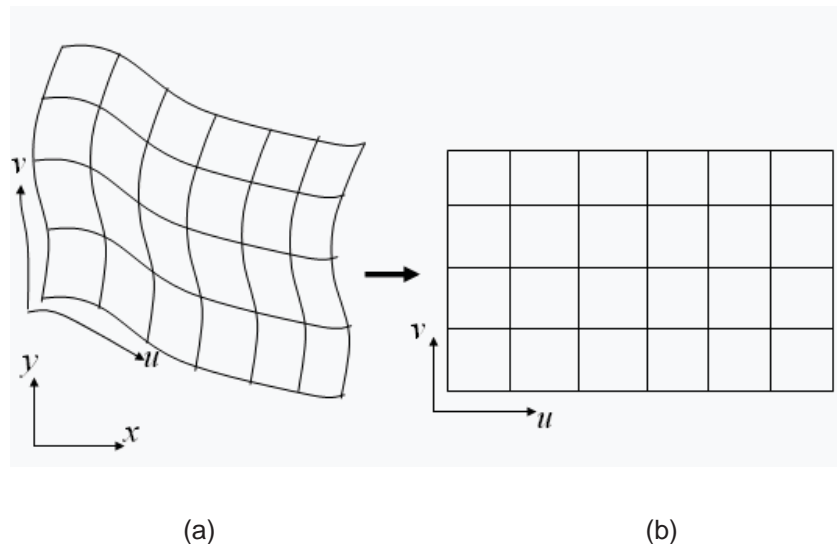



Figure 9.6: Mapping a surface to the parametric plane

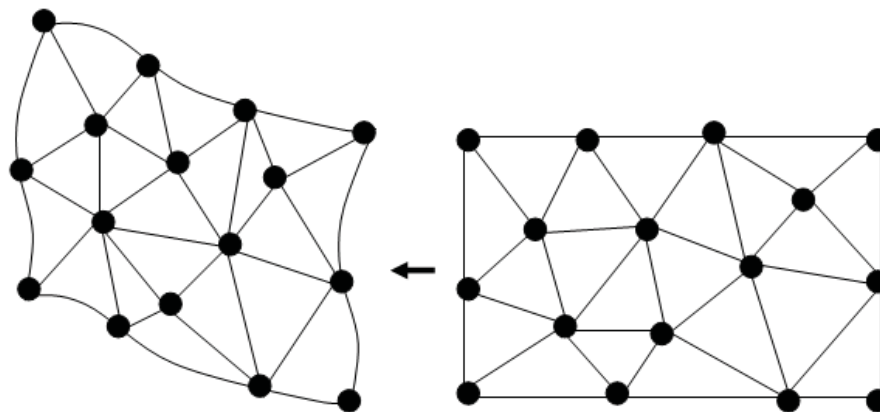



Figure 9.7 A mesh generated in the parametric plane, then mapped to the physical surface

Elements formed in the parametric space may not always form well-shaped elements when they are mapped to the physical surface. To resolve this problem, parametric surface mesh generation algorithms can do one of two things:

modify or re-parameterise the underlying parametric representation so there is a reasonable mapping from parametric space to the physical space;
modify the mesh generation algorithm so that stretched or anisotropic elements meshed in 2D will map back to well-shaped, isotropic elements in 3D;
Perform mesh smoothing operations on the mesh after it is mapped to the physical space.

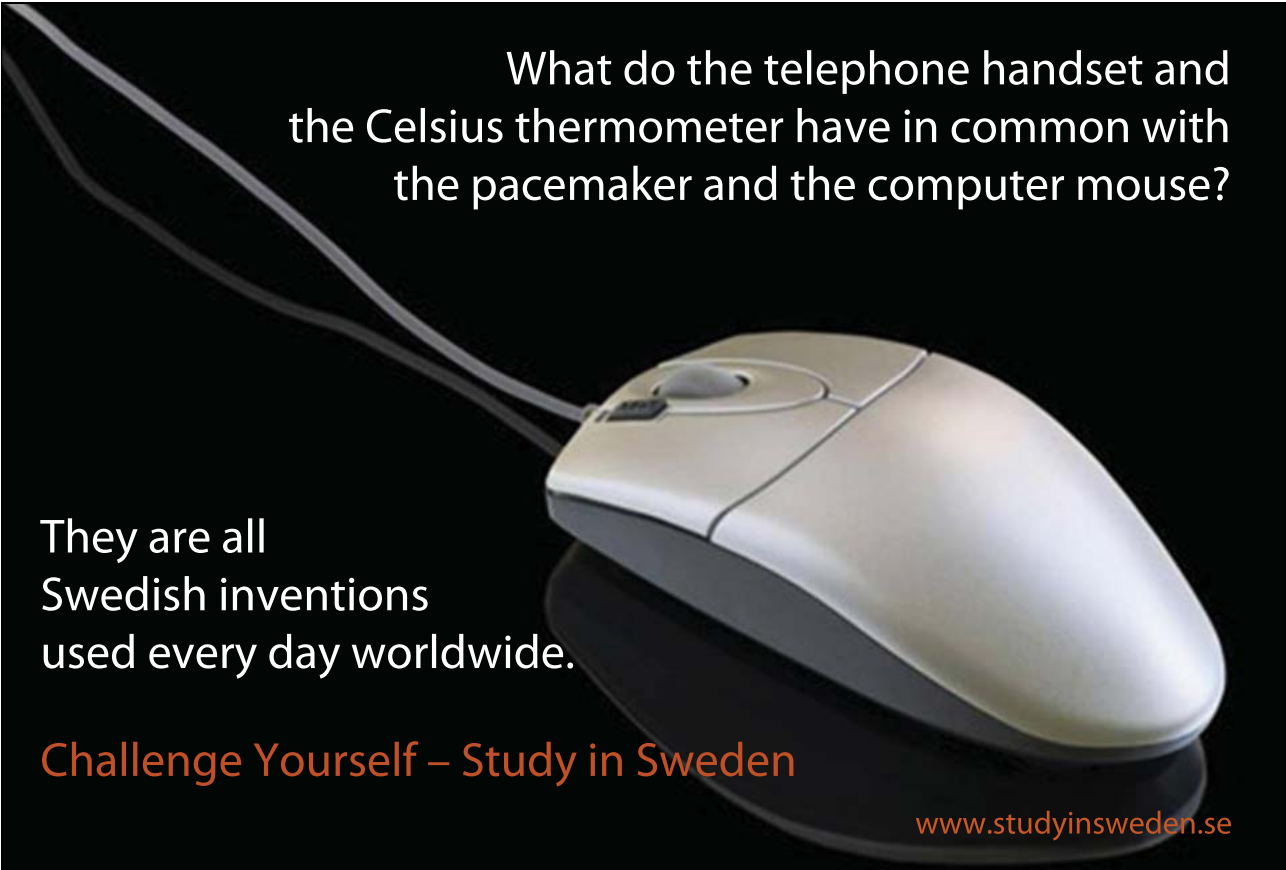
On the other hand, direct 3D surface mesh generators form elements directly on the geometry without regard to the parametric representation of the underlying geometry. In some cases where a parametric representation is not available or where the surface parameterization is very poor, direct 3D surface mesh generators can be useful. In these methods, a significant number of surface projections are required to ensure that new nodes generated in the meshing process are on the surface. Also of significance is the increased complexity of the intersection calculations required to ensure that elements on the surface do not overlap.

In the following sections, we will focus mostly on the mesh generation process in 2D, which is thought to be sufficient for explaining the basic principles of the mesh generation methods.

9.2.2 Advancing front method

This method is based on a bottom up idea of mesh generation. The concept is explained graphically in Figure 9.8 for a 2D surface. The method is used for generating triangular elements in 2D and tetrahedral elements in 3D. The first step is to discretise the domain boundaries. That is to divide the boundary into line segments following some functional distribution of size. This process is shown on Figure 9.8(a)

The discretised boundary is called the initial front. This is used to advance the triangles into the solution domain. The process follows the steps below:

A line segment is chosen as a base for a triangle. Typically, the shortest line segment in the front is used first.
A point is generated along the normal to the segment passing through mid point (see figure 9.9) such that the point is either at the same distance from the two ends of the segment or with a length specified by a local spacing function.
Intersection of the lines connecting the new point and the two ends of the segment with the front are checked. If there is no intersection, a valid triangle can be formed (Figure 9.8c)
If there is intersection with the front, either a series of other points are tried along the normal, closer to the edge, or a near by point from the front is used. The main criterion is to generate the least distorted triangle (Figure 9.8d).
Once a triangle is formed, the new edges are added to the front and the original edge is removed from the front. Thus the front is dynamically updated during the mesh generation process.
If a connection happens with an existing node in the front, more than one edge will be inside the mesh and can be eliminated from the front.

Steps 1 to 6 are repeated until the front has no edges remaining. In this case, the whole faces is meshed (figure 9.8f)

In 3D the boundary surfaces need to be triangulated first to form the initial front in. Each boundary surfaces will be represented using the parameterisation described in the previous section. Each surface is mapped into the parametric plane, triangulated and then the mesh is mapped onto the physical surface. This surface mesh forms the initial front.

From the initial front, steps are followed similar to those for generating triangles, but in this case, tetrahedral elements are generated. Conceptually, this is the same as in the 2D case, except the algorithms for checking intersections are more complicated.

The advancing front method has the advantage that the grid points and edges are generated at the same time allowing for good control on the mesh spacing. Also since the surface is triangulated first, this ensures that the method is boundary conforming. However, the method is relatively slow due to the large number of checks for intersection with the front during the mesh generation process.

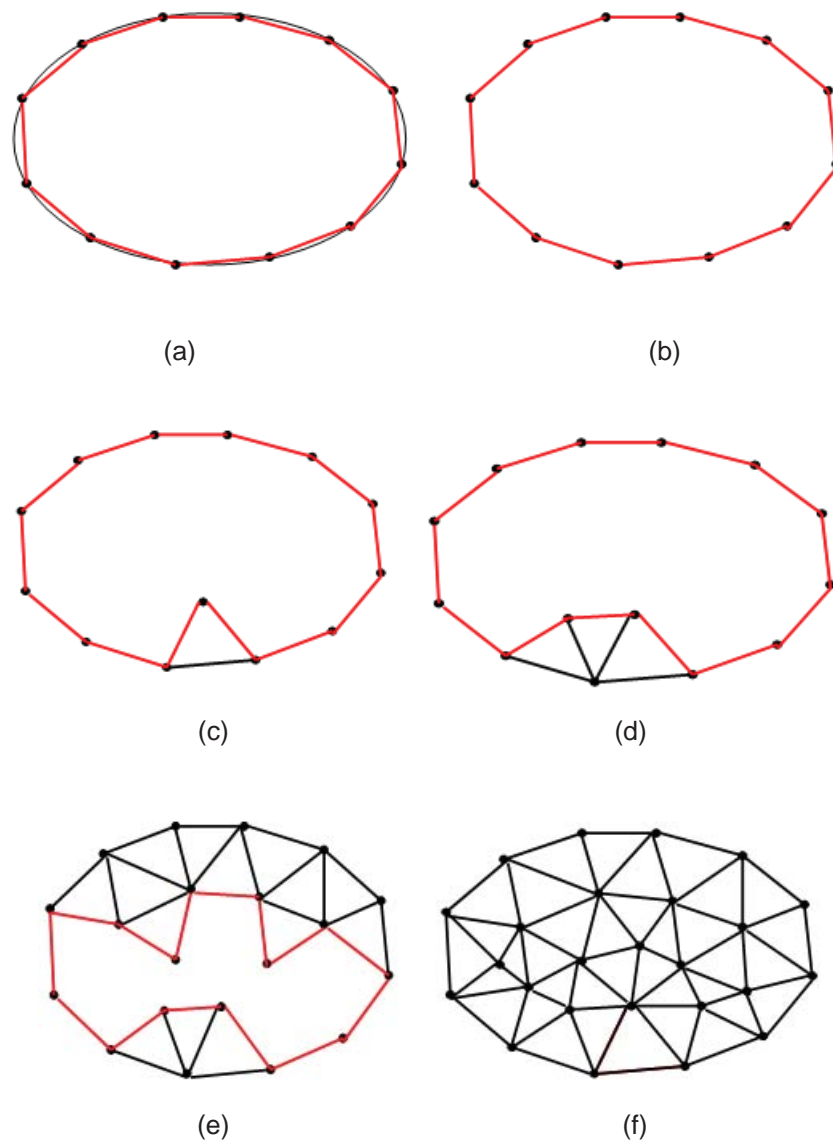



Figure 9.8: The advancing front method

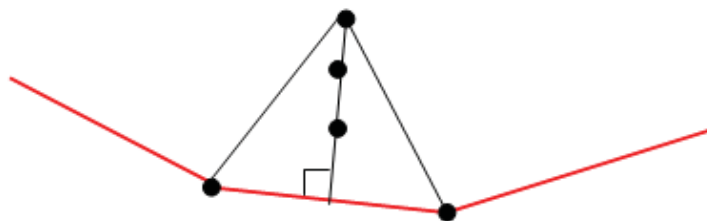



Figure 9.9: Generating a triangle on an edge

9.2.3 Delaunay triangulation

The Delaunay triangulation of a set of points has a well developed theory. The techniques used to generate the triangulation can obviously be used to generate unstructured meshes.

Given a set of discrete points in the plane, a Voronoi polygon about a point P is the region that is closer to, or as close to, P as any other point. The Voronoi or Dirichlet tessellation is made up of these Voronoi polygons and the Delaunay triangulation is the dual of the Dirichlet tessellation. Figure 9.10 illustrates these concepts.

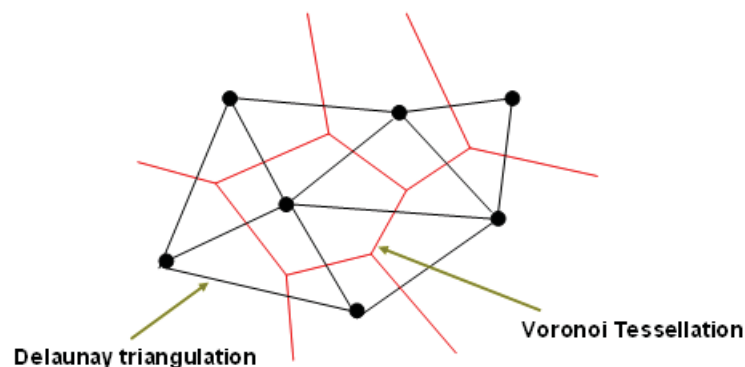



Figure 9.10: Delaunay triangulation and Voronoi Tessellation

The Delaunay triangulation has two properties that are useful in mesh generation:

No point is contained in the circumcircle of any triangle (that is the circle drawn to pass through the three points of the triangle). This property is used in Delaunay algorithms to ensure smoothness of the mesh.

In 2D, the Delaunay triangulation maximises the minimum angle of all triangular elements. Note that a good quality mesh requires minimising the maximum angle. In practical mesh generators, points are generated as well as triangles, so elements generated are any way of good quality. Unfortunately, this maximum minimum property is lost in 3D elements.

Delaunay triangulation process assumes that the points have been generated and the algorithm is the process of triangulating the existing points. Thus this process addresses only one half of the mesh generation problem. Thus some other algorithm needs to be used to generate points. One way of doing this is to use points from overlapping structured grids with some filtering or smoothing of the points.

Unlike the advancing front method, Delaunay method does not necessarily conform to the domain boundary. There are two ways however to force the Delaunay triangulation to conform to the boundary. In 2D it is possible to define a constrained triangulation. In this case, the pre-defined edges and included in the triangulation and the circumcircle property is modified to apply only to the points that can be seen from at least one node of the triangle, where the predefined edges are treated as opaque.

In 3D it is difficult to define the concept of constrained triangulation. Thus the boundary edges and faces are recovered using face and edge swapping processes, or by using more extensive re-triangulation to ensure that the mesh conforms to the boundary.

In addition to the above problems, poor quality grids may be generated and these need to be corrected. Despite these difficulties with Delaunay triangulation, it is some times preferred because it is the fastest method of generating unstructured grids.

9.2.4 Other methods

A variety of other methods have been developed for unstructured mesh generation, but they are all at the research stage with no widespread usage. In the following, we will outline some of these methods.

Paving and Plastering

Paving is the process of generating unstructured quadrilateral grids in 2D and plastering is the process of generating unstructured Brick elements in 2D. The two processes follow similar principles.

In the paving process, a boundary edge of a surface is discretised to line segments. All the line segments are advanced in the direction normal to the edge to produce a row of quadrilaterals. The process is repeated by selecting another edge at a time until the domain is covered. The process has some similarities with the advancing front and checks of intersections with the existing mesh need to be made at every stage. Figure 9.11 illustrates a paving algorithm for an arbitrary geometry.

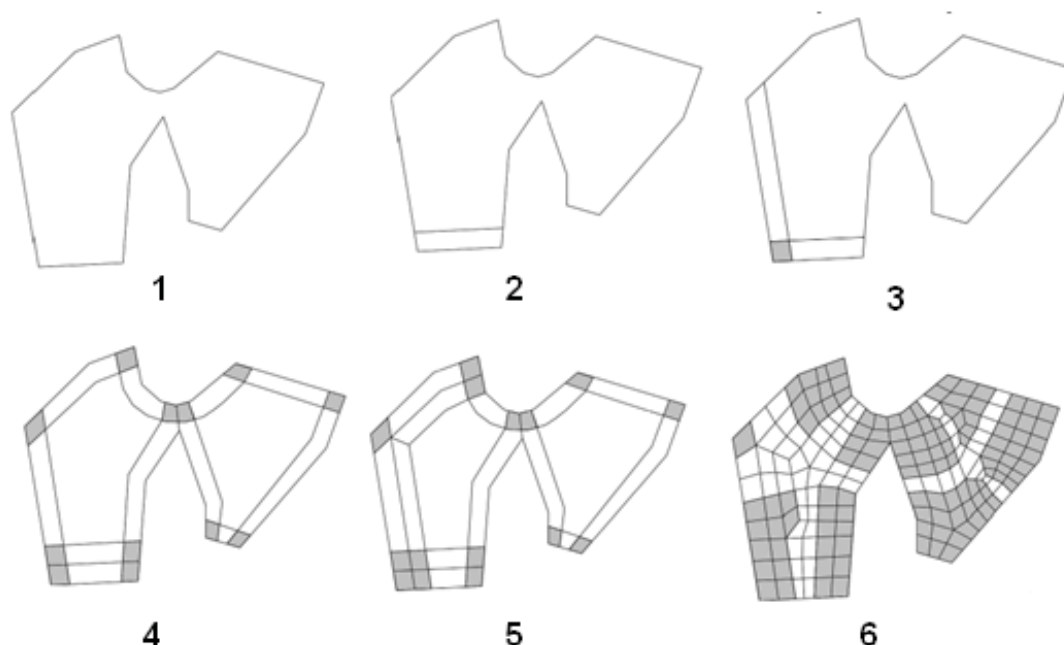



Figure 9.11 Paving

Octree method

In this method a coarse rectangular grid is overlaid on top of the geometry. The elements completely outside the computational domain are eliminated. The elements intersecting the boundary are successively split and any split element outside the geometry is eliminated. The process is continued until sufficient resolution is obtained near the boundaries. Finally points close to the boundary are moved onto the boundary to create the final mesh. An example of a mesh generated by this method is shown in Figure 9.12.

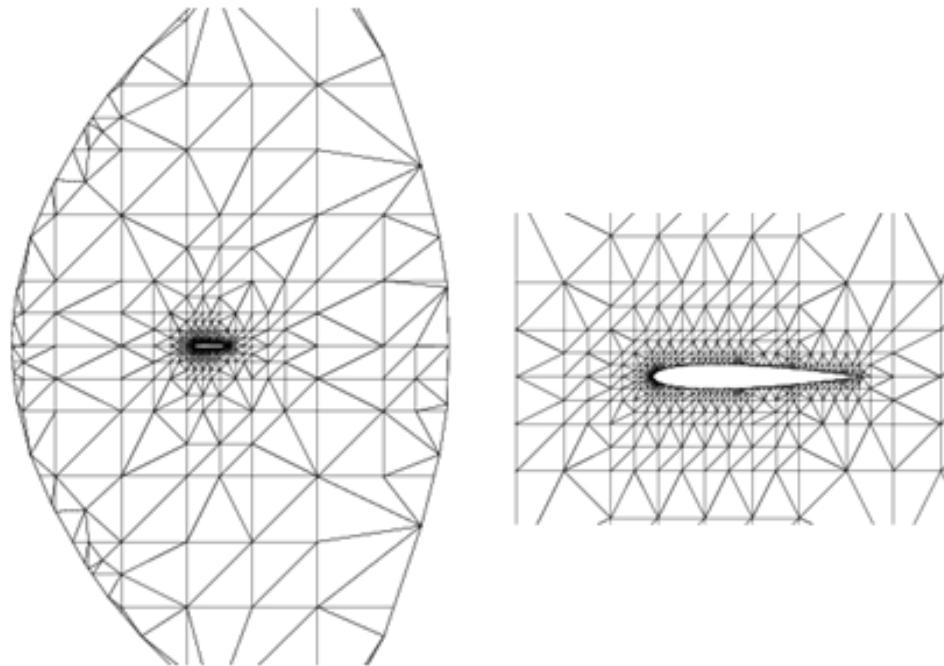



Figure 9.12: A mesh generated using Octree method

Semi-unstructured mesh generation

In some 3D geometries, the geometry might be complex in certain two coordinate directions, but slowly varying in the third direction. An example of this is turbomachinery blades, where the geometry is complex in the blade to blade direction, but less so along the blade. In this case, it may be possible to generate an unstructured mesh in the blade to blade direction allowing for capturing the blade curvature around the leading and trailing edge, and smoothly varying away from the blade. This mesh can then be copied in a structured manner along the blade and simply connected. An example of this type of mesh is shown in Figure 9.13. For more details, see Sbardella et al (1999)

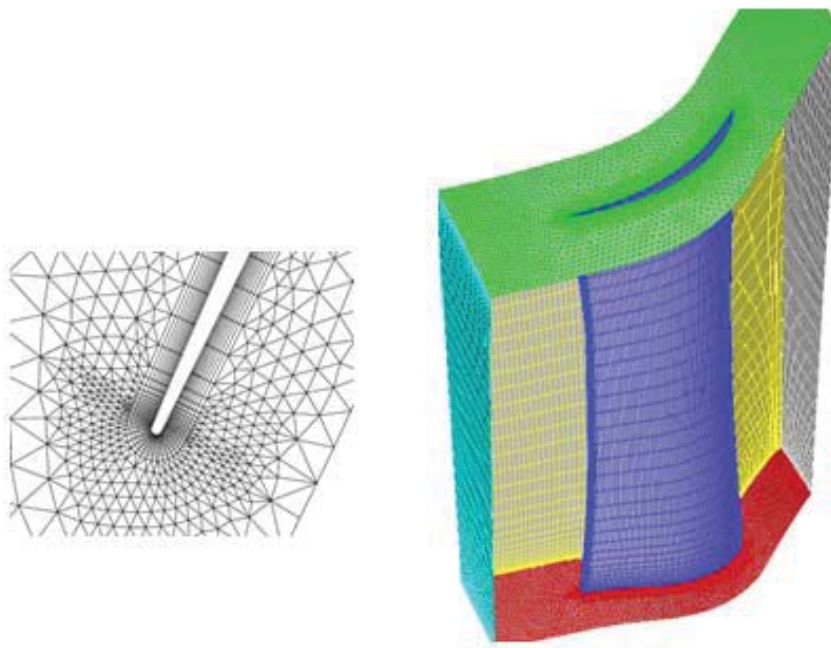



Figure 9.13: An example of a semi-unstructured mesh

References

Baldwin, B.S. and Barth, T.J. (1990), A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows, NASA TM 102847

Drikakis, D. and Rider, W. (2005) *High resolution methods for incompressible and low speed flow*, Springer-Verlag

Fletcher, C.A. J. (1991) *Computational Techniques for Fluid Dynamics*, 2nd edition. Springer.

Hirsch, C. (1998) *Numerical Computation of internal and external flows, Volume 1: Fundamentals of Numerical Discretisation*. Wiley

Sbardella, L., Sayma, A.I. and Imregun, M. (1999) "Semi-Unstructured Meshes for Axial Turbomachinery Blades", International *Journal of Numerical Methods in Fluids.* **32**(5), (1999), 569-584

Schlichting H. and Gersten, K.(2000) *Boundary Layer Theory*, 8th edition , Springer

Spalart, P.R., and Allmaras, S.R. (1992) A one-equation turbulence model for aerodynamic flows, *AIAA paper*, AIAA 1992-0438

Wilcox, D.C. (2004), *Turbulence Modeling for CFD*, ISBN 1-928729-10-X, 2nd Ed., DCW Industries, Inc

Zeinkiewicz, O. C. (1977) *The Finite Element Method*, 3rd ed. McGraw-Hill, London